User and Reference Manual



Copyright © 1998-2005 Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, StyleVision, Authentic, MapForce and ALTOVA are trademarks (registered in numerous countries) of Altova GmbH. Patent(s) pending.

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode, Inc. This software contains material that is ©1994-1998 Dundas Software Ltd., all rights reserved. The Sentry Spelling Checker Engine - ©2000 Wintertree Software Inc., STLport ©1999,2000 Boris Fomitchev, © 1994 Hewlett-Packard Company, ©1996,97 Silicon Graphics Computer Systems, Inc., ©1997 Moscow Center for SPARC Technology. Scintilla Copyright ©1998-2002 by Neil Hodgson <neilh@scintilla.org>. All other names or trademarks are the property of their respective owners.

ALTOVA

Altova StyleVision 2005 User Manual

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2005

© 2005 Altova GmbH 1998-2005 Altova GmbH - Patent(s) pending.

Printed in the United States of America

5 4 3 2 1

ISBN 1-933210-02-8

Table of Contents

1 Introduction 1

1.1 About stylesheets ... 2
1.2 Uses of StyleVision Power Stylesheets ... 3
1.3 StyleVision and Authentic View ... 4
1.4 StyleVision and output ... 6
1.5 StyleVision concepts and terminology ... 7
1.6 StyleVision features ... 8
1.7 About this documentation ... 14

2 Getting Started 16

2.1 Setting up StyleVision ... 17
2.2 StyleVision interface ... 18
2.2.1 Schema Window ... 20
2.2.2 Text and Block Style Windows ... 21
2.2.3 Document Design Tab ... 22
2.2.4 Authentic Editor View ... 23
2.2.5 XSLT-HTML Tab ... 24
2.2.6 HTML Preview ... 25
2.2.7 Menu Bar and Toolbar ... 26
2.3 Working with StyleVision ... 27
2.3.1 Document design process ... 28
2.3.2 Saving generated files ... 30

3 Tutorial 32

3.1 Setting up the StyleVision Power Stylesheet ... 33
3.2 Processing attribute values for output ... 36
3.3 Including an image ... 38
3.4 Processing element content for output ... 40
3.5 Formatting the components ... 44
3.6 Inserting a dynamic SPS table ... 48
3.7 Inserting a static SPS table ... 52

3.8 Inserting bookmarks and hyperlinks ... 56
3.9 Using Auto-Calculation ... 59
3.10 Using Conditional Templates ... 64
3.11 Creating the Table Footer ... 73
3.12 Text formatting with global templates and text state icons ... 77
3.13 Using Additional Validation ... 79
3.14 Completing the stylesheet ... 81

4 How To Use 86

4.1 XSLT and XPath versions ... 87
4.2 Namespaces ... 88
4.2.1 Namespaces and StyleVision ... 89
4.2.2 Schemas with a single namespace ... 91
4.2.3 Schemas with multiple namespaces ... 94
4.3 Global Templates ... 96
4.4 Rest-of-Contents ... 102
4.5 Adding Elements ... 103
4.6 Data-entry devices ... 106
4.6.1 Input fields ... 108
4.6.2 Combo boxes ... 109
4.6.3 Check boxes ... 111
4.6.4 Buttons ... 112
4.7 Lists ... 113
4.8 Tables ... 117
4.8.1 Creating static tables ... 120
4.8.2 Creating dynamic tables ... 121
4.8.3 Formatting static and dynamic tables ... 124
4.8.4 XML tables ... 126
4.9 Auto-Calculations ... 130
4.10 Conditional templates ... 133
4.11 Date Picker ... 136
4.12 Input Formatting ... 138
4.13 Parameters ... 145
4.14 Unparsed Entity URIs ... 147
4.15 StyleVision from the command line ... 148
4.16 Authentic View ... 150
4.16.1 Overview of the GUI ... 151
4.16.2 Authentic View toolbar icons ... 152

4.16.3 Authentic View main window ... 154
4.16.4 Authentic View context menus ... 156
4.16.5 Features and their usage ... 157
Tables in Authentic View ... *158*
Date Picker ... *165*
Define Entities ... *166*
Images in Authentic View ... *168*
Keystrokes in Authentic View ... *169*

5 Tips 172

5.1 The schema for the StyleVision Power Stylesheet ... 173
5.2 Handling components in the Design Document ... 174
5.3 Selective processing of siblings ... 176
5.4 Manually editing generated XSLT ... 177

6 Troubleshooting 180

7 HTML to XML Conversion 184

7.1 HTML-to-XML Tutorial ... 185
7.1.1 Opening the HTML ... 186
7.1.2 The starting schema and XML files ... 187
7.1.3 Creating element nodes ... 188
7.1.4 Importing an HTML list ... 192
7.1.5 Creating a section ... 194
7.1.6 Importing HTML tables ... 195
7.1.7 Creating schema attributes ... 197
7.1.8 Importing an HTML image ... 198
7.1.9 Converting HTML hyperlinks ... 199
7.1.10 Completing the import ... 200
7.1.11 Generating the output ... 201
7.2 HTML-to-XML Quick Reference ... 202
7.2.1 Importing the HTML file ... 203
7.2.2 Building the schema tree: 1 ... 204
7.2.3 Building the schema tree: 2 ... 206
7.2.4 Points to note ... 207

8 User Reference 210

8.1 Symbols and icons ... 211
8.2 Formatting ... 215
8.3 File ... 217
8.3.1 New... ... 218
8.3.2 Open... ... 219
8.3.3 Close ... 220
8.3.4 Import HTML File... ... 221
8.3.5 Assign Working XML File... ... 222
8.3.6 Edit Schema Definition in XMLSpy ... 223
8.3.7 Encoding... ... 224
8.3.8 Save Design ... 225
8.3.9 Save Design As... ... 226
8.3.10 Save Authentic XML Data... ... 227
8.3.11 Save Generated Files ... 228
8.3.12 Print Preview... ... 229
8.3.13 Print... ... 230
8.3.14 Most recently used files ... 231
8.3.15 Exit ... 232
8.4 Edit ... 233
8.4.1 Undo ... 234
8.4.2 Redo ... 235
8.4.3 Find ... 236
8.4.4 Find Next ... 238
8.4.5 Edit Stylesheet Parameters... ... 239
8.4.6 Select All ... 240
8.5 Insert ... 241
8.5.1 Image... ... 242
8.5.2 Paragraph ... 243
8.5.3 Format ... 244
8.5.4 Bullets and Numbering ... 245
8.5.5 Horizontal Line ... 246
8.5.6 Contents ... 247
8.5.7 Rest of Contents ... 248
8.5.8 Date Picker ... 249
8.5.9 Page ... 250
8.5.10 Bookmark ... 251
8.5.11 Hyperlink ... 252
8.5.12 Auto-Calculation ... 253
8.5.13 Condition ... 255
8.6 Table ... 256

8.6.1 Insert Table... 257
8.6.2 Delete Table 258
8.6.3 Append Row 259
8.6.4 Append Column 260
8.6.5 Insert Row 261
8.6.6 Insert Column 262
8.6.7 Delete Row 263
8.6.8 Delete Column 264
8.6.9 Join Cell Left 265
8.6.10 Join Cell Right 266
8.6.11 Join Cell Below 267
8.6.12 Join Cell Above 268
8.6.13 Split Cell Horizontally 269
8.6.14 Split Cell Vertically 270
8.6.15 View Cell Bounds 271
8.6.16 Table Properties 272
8.6.17 Vertical alignment of cell content 273
8.7 Authentic 274
8.7.1 Node Settings... 275
8.7.2 Text State Icons... 278
8.7.3 CALS / HTML Tables... 280
8.7.4 Assign Template XML File... 282
8.7.5 Auto-add Date Picker 284
8.7.6 Table 285
8.7.7 Markup 286
8.7.8 Define Entities 287
8.7.9 Validate XML 288
8.8 HTML Import 289
8.8.1 Import HTML file... 290
8.8.2 Add to Parent 291
8.8.3 Insert 292
8.8.4 Append 293
8.8.5 Settings... 294
8.9 Properties 295
8.9.1 HTML Page... 296
8.9.2 Table... 297
8.9.3 Bullets and Numbering... 298
8.9.4 Predefined Format Strings 299
8.9.5 Selected object... 301
8.10 Tools 302

8.10.1 Spelling... 303
8.10.2 Spelling options... 304
8.10.3 Customize... 307
8.10.4 Options... 312
8.11 Help 313
8.11.1 Table of contents... 314
8.11.2 Index... 315
8.11.3 Search... 316
8.11.4 Registration... 317
8.11.5 Order form... 318
8.11.6 Support Center... 319
8.11.7 FAQ on the web... 320
8.11.8 Components download... 321
8.11.9 StyleVision on the Internet... 322
8.11.10 About StyleVision... 323
8.12 Context menus 324
8.12.1 Change to 325

9 Appendices 328

9.1 OS and Memory Requirements 329
9.2 Electronic Software Distribution 330
9.3 License Metering 331
9.4 Copyright 332
9.5 Software Product License 333

Index 343

Chapter 1

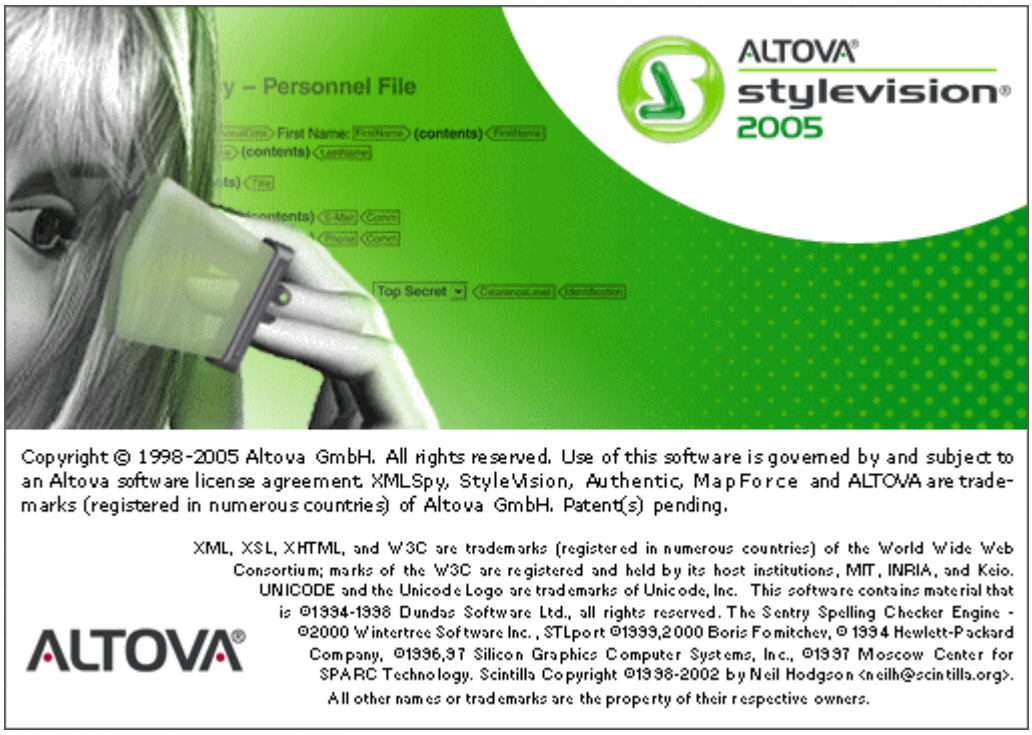
Introduction

1 Introduction

Altova® StyleVision® 2005 Professional Edition is an intuitive visual editing tool that enables Web developers and document system managers to create StyleVision Power Stylesheets and to generate XSLT stylesheets from them. The key features of StyleVision are grouped under the following main areas of functionality:

- The creation of StyleVision Power Stylesheets, which are used to control the display of, and entry of data into, XML documents in the Authentic View of Altova products.
- The generation of XSLT stylesheets from StyleVision Power Stylesheets. These enable transformations of source XML documents into outputs such as HTML.

Altova StyleVision 2005 also enables you to import an HTML file and convert it to an XML file.


ALTOVA
stylevision
2005
Copyright © 1998-2005 Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, StyleVision, Authentic, MapForce and ALTOVA are trademarks (registered in numerous countries) of Altova GmbH. Patent(s) pending.
XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode, Inc. This software contains material that is ©1994-1998 Dundas Software Ltd., all rights reserved. The Sentry Spelling Checker Engine - ©2000 Wintertree Software Inc., STLport ©1999,2000 Boris Fomitchev, © 1994 Hewlett-Packard Company, ©1996,97 Silicon Graphics Computer Systems, Inc., ©1997 Moscow Center for SPARC Technology. Scintilla Copyright ©1998-2002 by Neil Hodgson <neilh@scintilla.org>.
All other names or trademarks are the property of their respective owners.
ALTOVA

Writing even the simplest XSLT stylesheets by hand is a difficult task. It requires an understanding of XSLT elements, the XPath language, and complicated rules-based document processing models. By abstracting away the underlying error-prone syntax, **Altova® StyleVision® 2005** enables you to write stylesheets using considerably less time. **Altova® StyleVision® 2005** supports both XSLT 1.0 and XSLT 2.0.

1.1 About stylesheets

XSLT stylesheets are used by XSLT engines to transform one XML document into another XML, HTML, or text document. In the transformation process, according to instructions in the XSLT file, the XSLT processor selects content from the XML source, and places it in the output document. The selected content is usually placed as element content or as an attribute value. If the output document is intended for rendition on some medium, or is part of a processing chain that produces a document to be rendered, then the XSLT stylesheet can also be used to add presentation properties to the output document. The instructions for content selection, processing, placement, and formatting are all contained in the XSLT stylesheet.

StyleVision enables you to create a special type of XSLT-based stylesheet, called a **StyleVision Power Stylesheet (or SPS file)**. A StyleVision Power Stylesheet has two uses:

- It enables XML documents to be edited in Authentic View
- It can be used to generate XSLT stylesheets. Therefore, StyleVision can be used as a graphical, XSLT builder application

You do not need to know XSLT to be able to use StyleVision. Most of the XSLT constructs in the StyleVision Power Stylesheet are created with mechanisms in the StyleVision interface, and you will be able to construct simple StyleVision Power Stylesheets after learning to use these mechanisms. In order to build more sophisticated stylesheets with StyleVision, you should have a knowledge of XPath.

StyleVision supports both XSLT 1.0 and XSLT 2.0. The XSLT 2.0 specification is considerably more powerful than XSLT 1.0, and with XSLT 2.0 you can access and manipulate XML documents with concise and efficient mechanisms. In StyleVision, you can select whether you wish to use XSLT 1.0 or XSLT 2.0 for individual StyleVision Power Stylesheets. Depending on your selection, the appropriate functionality will automatically be made available in the design interface and transformations will automatically be made with the corresponding XSLT processor (XSLT 1.0 or XSLT 2.0).

Note: Since much of the power of XSLT 2.0 derives from XPath 2.0 features, you should be familiar with XPath 2.0 in order to maximize the utility of XSLT 2.0 in StyleVision.

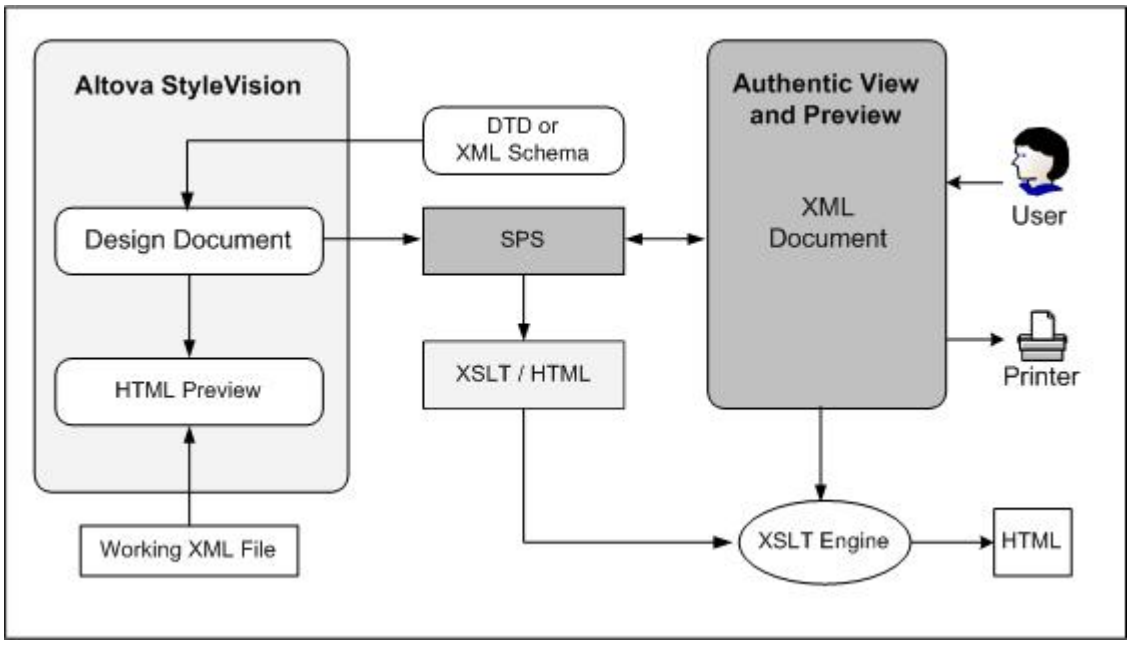
© 2005 Altova GmbH

1.2 Uses of StyleVision Power Stylesheets

The design document that you create in StyleVision is saved as a **StyleVision Power Stylesheet**, or **SPS file** (so called because of its `.sps` file extension). A StyleVision Power Stylesheet can be used for the following purposes:

- To display an XML file in Authentic View. Authentic View is a graphical user interface that allows users unfamiliar with XML to enter content in the XML document. You should therefore make your StyleVision Power Stylesheet as user-friendly as possible. The WYSIWYG Authentic View of an XML document can be printed directly to a printer
- To generate an XSLT stylesheet to obtain HTML output from the source XML

The illustration below shows how StyleVision, StyleVision Power Stylesheets, and Authentic View can be used in an XML environment.

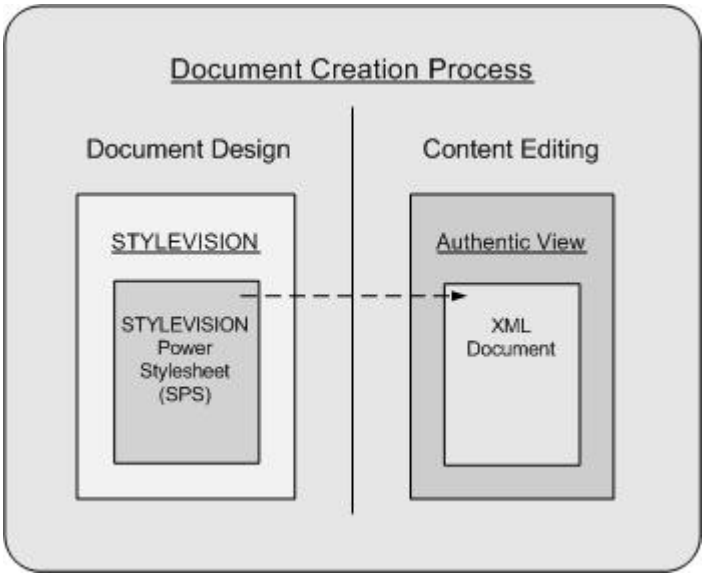


Note: Altova XMLSpy 2005 and Altova Authentic 2005 Desktop Edition have built-in Altova XSLT processors. So XML documents created in the Authentic View of these products can be processed with a StyleVision-generated XSLT stylesheet directly from within the interface. Additionally, the Enterprise and Professional editions of XMLSpy 2005 also include support for external XSLT processors and FO processors. This means that, from within XMLSpy 2005 Enterprise and Professional editions, you can use either the built-in XSLT processor or an XSLT processor of your choice to obtain HTML output.

1.3 StyleVision and Authentic View

The StyleVision Power Stylesheet you create with StyleVision is used to control the display of an XML document in Authentic View (a document view available in Altova products; see Authentic View in Altova Products). With Authentic View, users who are unfamiliar with XML can easily enter XML document content without needing to learn XML.

A document creation process that involves Authentic View consists of two separate stages: **document design** and **content editing**. The StyleVision Power Stylesheet that is created (with StyleVision) in the document design stage is linked to the XML document in Authentic View and controls the way the XML document is processed and displayed.



Document design with StyleVision
In the document design stage, a developer—typically an XML-knowledgeable person—uses Altova's StyleVision to create an StyleVision Power Stylesheet.

The StyleVision Power Stylesheet basically processes the XML document to produce a display of the XML document. Most importantly, however, and unlike typical XSLT stylesheets, which are used to process **finished XML documents**, the StyleVision Power Stylesheet also provides mechanisms for the Authentic View user to **input data into the XML file**. Authentic View, therefore, provides users the opportunity to input XML data and to instantaneously preview the output of the XML document.

Not only does the StyleVision Power Stylesheet control data input to an XML document. It also defines how data should be processed and output. These processing definitions are used to produce the end-outputs: the Authentic View display and HTML files.

Content editing with Authentic View
Authentic View is an interface for editing an XML document that is linked to a StyleVision Power Stylesheet.

- When a new XML document is created, it can be assigned a StyleVision Power Stylesheet, and the document can then be edited in Authentic View using the document template and controls specified in the StyleVision Power Stylesheet.
- Existing XML documents can be assigned a StyleVision Power Stylesheet (which must be based on the same schema as the XML file) at any time, and can be edited subsequently in Authentic View.

 © 2005 Altova GmbH

The user of Authentic View is not expected to be knowledgeable about either XML or the schema being used for the document. The document display in Authentic View should make content editing as easy and non-technical as possible. It is, therefore, the task of the person who designs the StyleVision Power Stylesheet to produce a user-friendly Authentic View display.

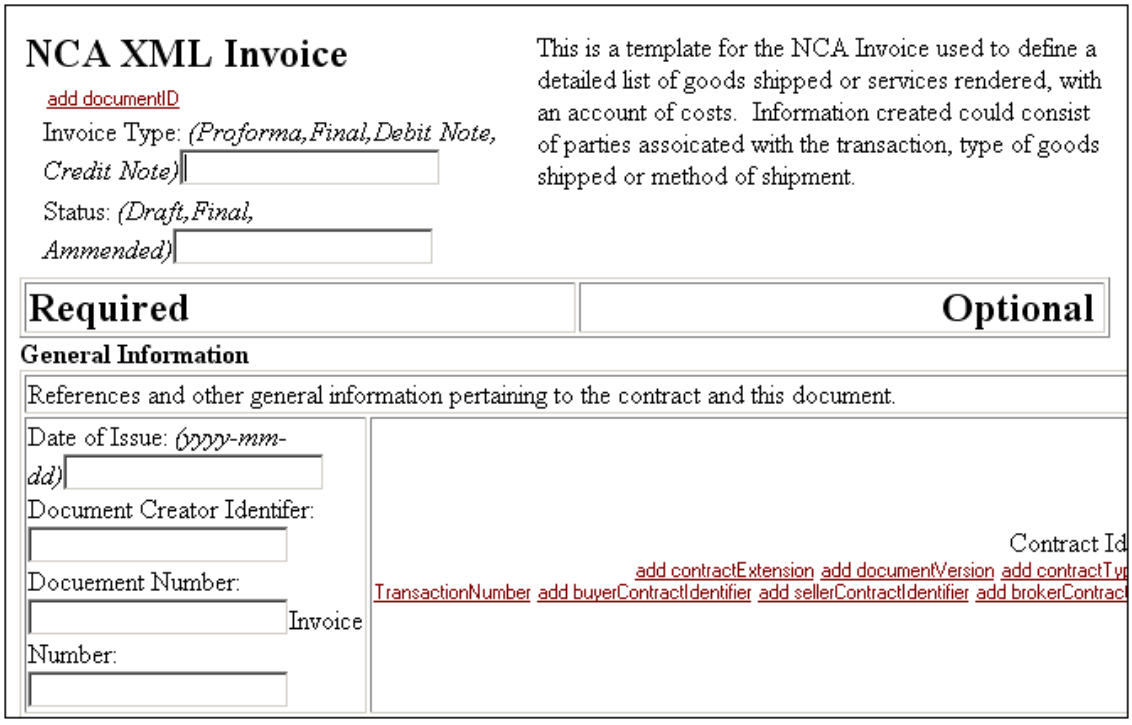
For detailed information about using Authentic View, see the Authentic View documentation in the user manual of XMLSpy 2005 or Authentic 2005 Desktop Edition.

Authentic View in Altova Products
Authentic View is a WYSIWYG view that enables users to edit XML documents in a GUI. Authentic View is available as a view in the following Altova products:

- Altova® XMLSpy 2005® (Enterprise, Professional, and Home editions)
- Altova® Authentic® Desktop Edition
- Altova® Authentic® Browser Edition
- Altova® Altova StyleVision 2005 (as a preview window called Authentic Editor, in which you can edit the XML file)

The Authentic View Interface and Features are explained in the relevant sections of the Authentic View documentation. For a full description of Authentic View, see the Authentic View Tutorial in the User Manual of XMLSpy 2005 or Authentic 2005 Desktop Edition, which are both available at the Altova website.

StyleVision Power Stylesheets for standard industry schemas
Altova's Authentic View package includes StyleVision Power Stylesheets for a number of standard industry schemas. Users can therefore immediately create an XML document based on a standard schema in Authentic View. The figure below shows a partial Authentic View of the NCA Invoice.


NCA XML Invoice
add documentID
Invoice Type: (Proforma,Final,Debit Note, Credit Note)
Status: (Draft,Final, Ammended)
This is a template for the NCA Invoice used to define a detailed list of goods shipped or services rendered, with an account of costs. Information created could consist of parties assoicated with the transaction, type of goods shipped or method of shipment.
Required
Optional
General Information
References and other general information pertaining to the contract and this document.
Date of Issue: (yyyy-mm-dd)
Document Creator Identifer:
Docuement Number:
Invoice Number:
Contract Id
add contractExtension add documentVersion add contractTyp
TransactionNumber add buyerContractIdentifier add sellerContractIdentifier add brokerContract

You can easily customize any of the supplied standard industry StyleVision Power Stylesheets, which are available in the **Examples/IndustryStandards** folder of your application folder.

1.4 StyleVision and output

In StyleVision, you can generate XSLT stylesheets for output to HTML from your StyleVision Power Stylesheet. See figure in Uses of StyleVision Power Stylesheets for an overview.

Presentation properties (formatting, layout, hyperlinking, etc) in the StyleVision Power Stylesheet are translated into the corresponding HTML markup. Essentially, therefore, one design document (created in Design View) is translated into the markup for the XML document displayed in Authentic View and for the HTML output. These transformations from XML source to marked-up output documents are achieved with XSLT stylesheets. You can see output previews in the HTML Preview Window.

Note: Since Authentic View is a display of an XML file, and since the Authentic View display accepts input for the XML, some components in the Authentic View display do not have corresponding input devices in the HTML output. Such components could be data-entry devices, such as text fields, combo boxes, and check boxes. If these components are used in the Design Document, a suitably correspondent component appears in the HTML output.

XSLT for HTML output
For HTML output obtained by using a StyleVision Power Stylesheet-generated XSLT, the following points should be noted:

- The formatting and layout of the HTML will be identical to the HTML Preview of StyleVision and near-identical to the Authentic View of the XML document.
- Data-input devices (text input fields, check boxes, etc) in the HTML file do not allow input. These data-input devices are intended for XML data input in Authentic View and, though they are translated unchanged into the graphical HTML equivalents, they cannot be used for data-entry in the HTML document.

Using the command line to generate output
You can use the command line to call StyleVision and generate various output files. For details, see StyleVision from the command line.

© 2005 Altova GmbH

1.5 StyleVision concepts and terminology

The following concepts and definitions are important for an understanding of how StyleVision Power Stylesheets and Authentic View work. The terms listed below are used frequently in this documentation. Some are specific to Alt ova's StyleVision and Authentic View. Others are terms used in the XML, XML Schema, XSLT, and XPath specifications or terms commonly used in discussing XML subjects.

StyleVision concepts
The following concepts and definitions are specific to StyleVision. This list is complemented by a diagrammatic representation of the related parts of the StyleVision working environment.

- A **StyleVision Power Stylesheet** is a **.sps file** created with **StyleVision**. It is built around the XSLT programming language
- A StyleVision Power Stylesheet is based on a particular **schema** (DTD or XML Schema)
- A **Working XML File** is assigned to a StyleVision Power Stylesheet in StyleVision in order to preview the output of an XML document in StyleVision. It also resolves ambiguity about the document element if such exists (see Troubleshooting)
- A **Template XML File** is an XML file that provides the starting data of the XML Document that is used as the Authentic View template. It must be conformant with the schema on which the StyleVision Power Stylesheet is based. A Template XML file is assigned to a StyleVision Power Stylesheet in StyleVision
- An XML file can be assigned a StyleVision Power Stylesheet at any time in a Authentic View product
- Once a StyleVision Power Stylesheet has been assigned to an XML document, the latter can be opened in Authentic View and edited; it is an XML document like any other.

Note: For a listing and description of StyleVision GUI components, see StyleVision interface.

General XML, XML Schema, XSLT, XPath terms
A number of XML, XML Schema, XSLT, and XPath terms appear frequently in this documentation. Terms that are important in the StyleVision context, or those that may be unclear or ambiguous, are listed below.

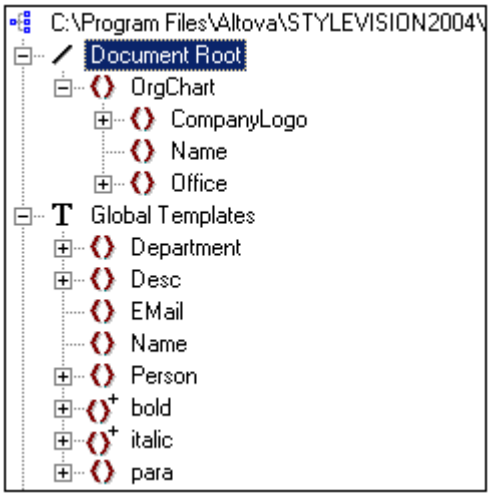
- *DTD, XML Schema, schema:* We use the word *schema* (lowercased) to denote any and all types of validation grammars. The most commonly used schemas are *XML Schemas* (capitalized) and *DTDs*.
- *Root Node, Document Node, Document Root:* In XPath 1.0, the node that contained the entire XML document was called the Root Node. In XPath 2.0, the Root Node is called the Document Node. In StyleVision, the term Document Root is used. In this documentation, these three terms are used according to context. They refer to the same node. This node is an abstract node that is part of the abstract data model of the XML document; there is no corresponding node in the actual XML document.
- *Document Element, Root Element:* The Document Node contains a single root element, the Document Element. In some literature, this element node is called the Root Element. In this documentation, we use the term Document Element. This element is the main container element of the XML document, and in the data model occurs immediately under the Document Node.
- *Root Template, Global Template:* A Root Template matches the Document Node. In StyleVision, when you drag elements from the schema tree into the Design Document, you are creating or modifying the Root Template. Only one Root Template is created in StyleVision. A Global Template can be created for any element declared in a DTD and for any *global* element declared in an XML Schema.

1.6 StyleVision features

The following is a list of major features supported in StyleVision.

Choice of XSLT and XPath language versions
StyleVision Power Stylesheets are built upon the XSLT and XPath languages, and StyleVision supports both XSLT 1.0 (which works in conjunction with XPath 1.0) and XSLT 2.0 (which works with XPath 2.0). When you create a StyleVision Power Stylesheet, you can select the XSLT language version (1.0 or 2.0) in which that StyleVision Power Stylesheet must be written. The selection ensures (i) that the correct language version features are displayed for that StyleVision Power Stylesheet in the GUI, and (ii) that the correct processor (XSLT 1.0 or XSLT 2.0) is used for internal XSLT validations (e.g. XPath validations) and transformations (e.g. for Previews).

Document structure
The **Root Template** is the template that defines the structure of the document. It enables you to control the structure of the document by selecting nodes from the schema tree (shown in the Schema Window below under `/ Document Root`) and placing them in the document structure in the Design Window. You can move nodes around in the design document, and thus control the structure of the result document.



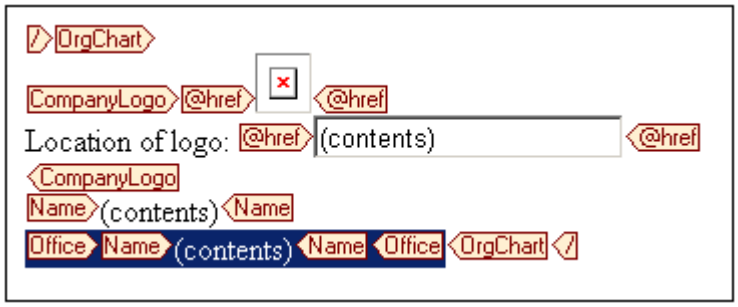
A **global template** can be created for each global element type (XML Schema elements defined as global elements and all DTD elements). Using a global template is convenient because it allows you to globally define rules for processing an element type. You do not have to re-create the same rules at each location in the design document where an element type occurs. When an element for which a global template is defined is placed in the Root Template, you can choose between using the global template and using a local template for that occurrence.

The Root Template gives you control over the structure of the result document. Global templates allow you to write rules for elements, the locations and frequency of which in the instance document you cannot know in advance. For example, you cannot know, before a paragraph is written, whether it will contain italics. So you cannot place a rule for processing italic elements in the Root Template, the structure of which corresponds to the output document structure. Global templates also give you efficiency. For example, if a title element is to occur several times in a document, a single global template can be used for all title elements. You can use an appropriate mix of Root Template and global templates in your StyleVision Power Stylesheet.

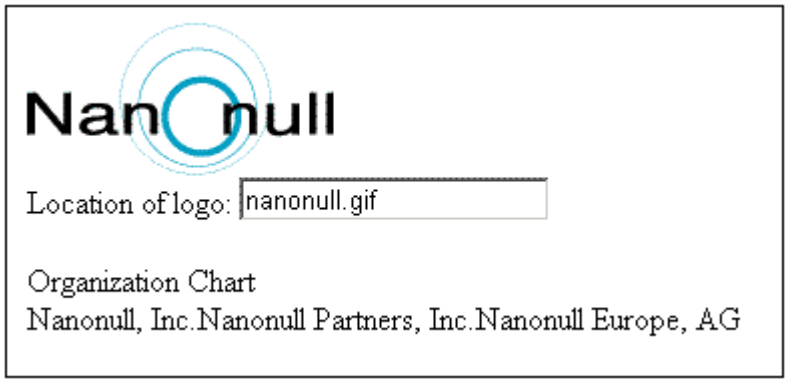
Looping through a nodeset
A common feature of XML documents is the repeating data structure. For example, an office department typically has several employees. The data for each of these employees is stored in

 © 2005 Altova GmbH

a data structure which is repeated for each employee. Similarly, the data for each department in the office is stored in a repeating data structure. In the StyleVision Power Stylesheet, the processing for each such data structure is defined in a single template, and the rules of this template are applied to each node in the nodeset in turn.



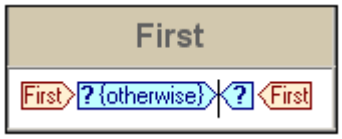
Creating a template for `//Office/Name` causes the names of all three offices to be output, as shown below.



Conditional templates
Conditions can be attached to a template. This allows you to specify alternative processing for different XPath-based conditions. For example, a condition can specify that a template apply only to the first node in a nodeset, or that nodes with numerical values above a certain value be processed differently than those with a value equal to or less than that value.

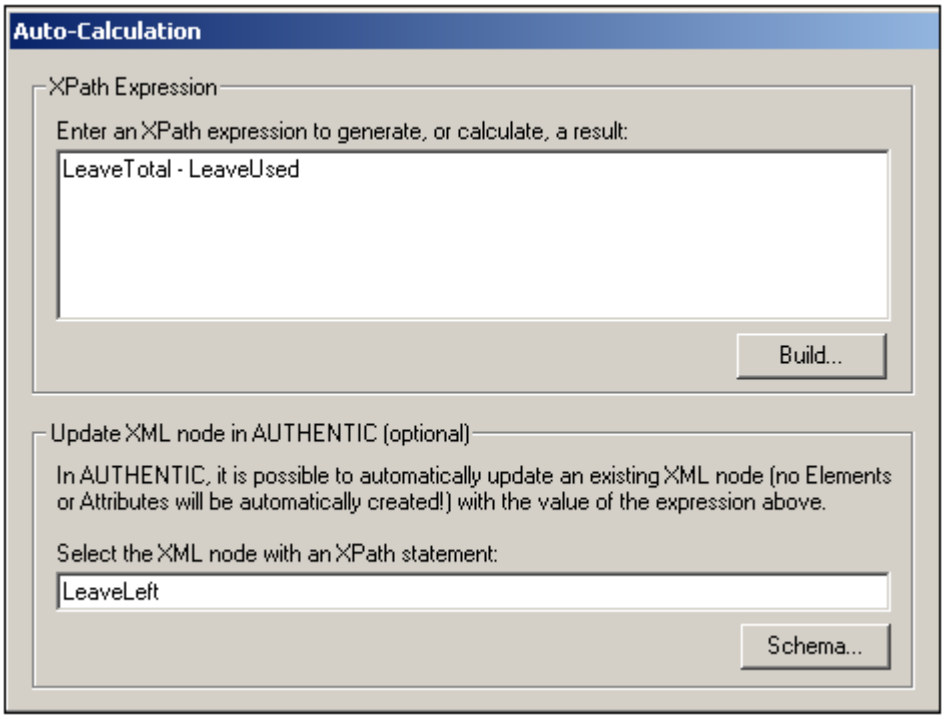


The screenshot above shows a condition inside the `First` field. The condition checks whether the number of shares (value of the `Shares` element) exceeds the value `0`. If it does, the content of the `First` field is displayed in bold. A second condition within the template (shown below) is an Otherwise template, which causes no output to be displayed. So if the value in the `Shares` field is `0` or does not exist, then the content of the `First` element is not displayed (or, alternatively, could be displayed in roman).



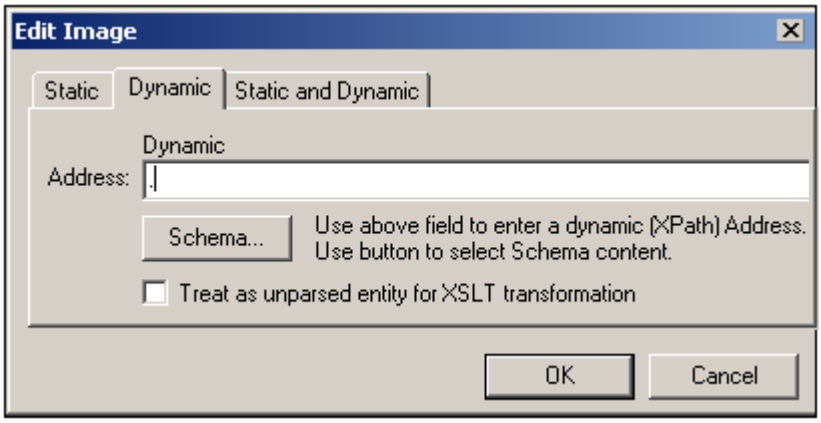
Auto-Calculations
An Auto-Calculation evaluates an XPath expression and displays the result in Authentic View.

Optionally, the result can be entered as XML node content. This is useful, for example, for totalling amounts in a column, or for counting the total number of employees in an office, or for any mathematical operation involving node content in various parts of the document.


Auto-Calculation
XPath Expression
Enter an XPath expression to generate, or calculate, a result:
LeaveTotal - LeaveUsed
Build...
Update XML node in AUTHENTIC (optional)
In AUTHENTIC, it is possible to automatically update an existing XML node (no Elements or Attributes will be automatically created!) with the value of the expression above.
Select the XML node with an XPath statement:
LeaveLeft
Schema...

Static and dynamic text
Static or dynamic text can be inserted anywhere in the document. Static text is text that is contained in a template. It is output as invariable text, such as labels and table-column headers, in the result document. Dynamic text, on the other hand, is text that comes from the XML document. It changes with each node in a nodeset and when the XML document is edited.

Static and dynamic images
A static or dynamic image can be inserted anywhere in the document. The location of static images is specified in the StyleVision Power Stylesheet, while the location of dynamic images is specified as node content in the XML document. In the StyleVision Power Stylesheet, the location of a dynamic image is referenced via the node containing it. Thus, in a repeating data structure, the image referenced can be different for each node in the nodeset, and depends on the image location information contained in that node.



 © 2005 Altova GmbH

Inserting static-and-dynamic images (where part of the address is static and part is dynamic) is also possible.

Static and dynamic hyperlinks
A static or dynamic hyperlink can be inserted anywhere in the document. As with static and dynamic images, hyperlink addresses can be contained in either the StyleVision Power Stylesheet or in the XML document, or partly in both.

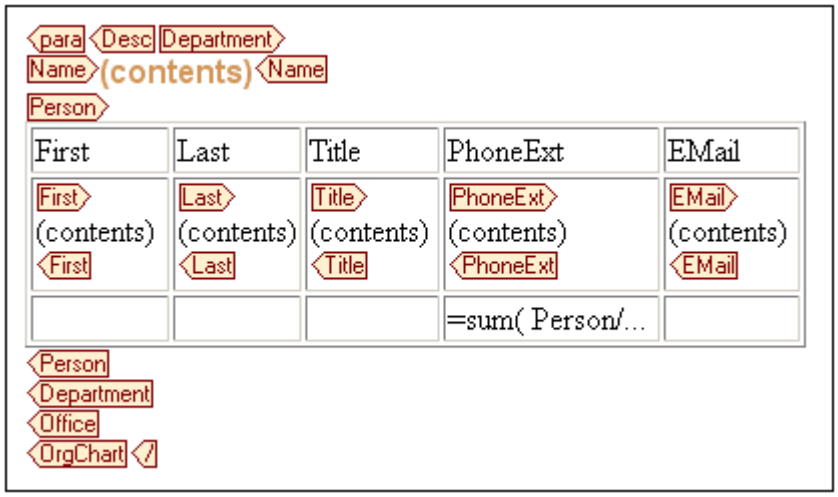
Static bookmarks
A static named bookmark (or anchor) can be inserted anywhere in the document, and can be referenced by a hyperlink.

Static and dynamic tables
A static or dynamic table can be inserted anywhere in the document.

A static table is used to generate a table in the result document in which cell content comes from unrelated nodes in the document. In the static table shown below, nodes from anywhere in the schema can be dropped into any table cell.



A dynamic table (below) generates a table in the result document for a repeating data structure: the child nodes form one dimension of the table while the element occurrences form the other dimension.

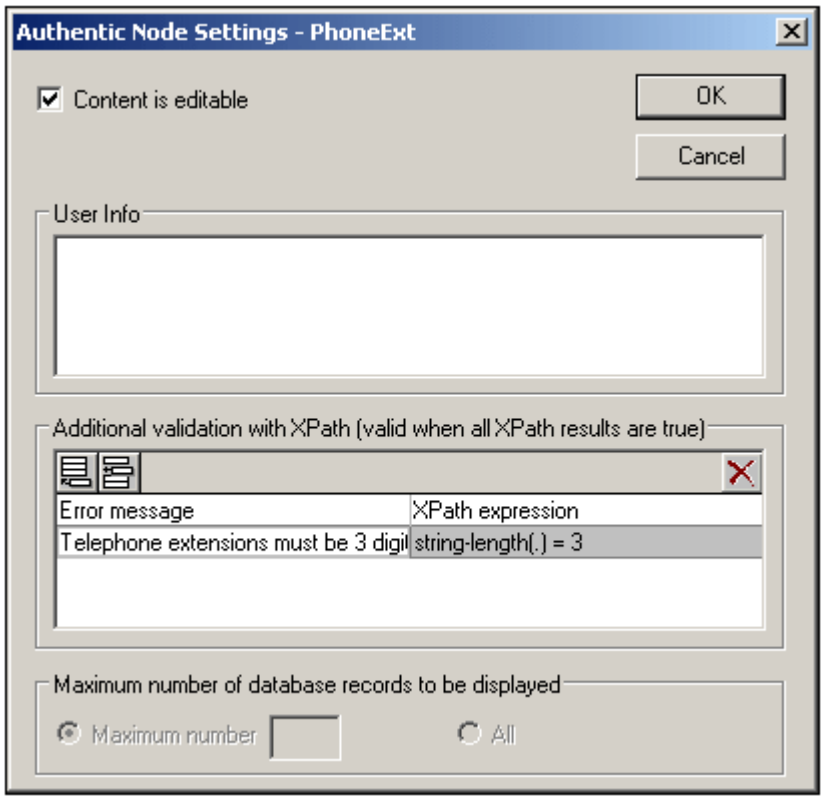


Both static and dynamic tables can be re-structured and re-formatted as required.

Static and dynamic lists
A static or dynamic list can be inserted anywhere in the document. A static list is a list in which the list items are created directly in the StyleVision Power Stylesheet. A dynamic list is one in which it is defined that multiple instances of a single element in the XML document are to be created as the list items of a list.

Additional Validation
One or more XPath expressions can be set for each node. The content of the node can be

evaluated against this expression and a suitable error message displayed in Authentic View in the event that the XPath is evaluated as false. In the example below, Additional Validation is set for the node `PhoneExt`. If the value of the `PhoneExt` node is a string of length not equal to three, then a validation error is generated and the message entered in the Error message filed is displayed.



Parameters
Parameters can be declared for the stylesheet. This enables values to be passed to the stylesheet from the command line at processing time. See Parameters for details.

Command line transformations and file-generation
StyleVision can be called from the command line to silently execute XSLT transformations and generate output files. See StyleVision from the command line for details.

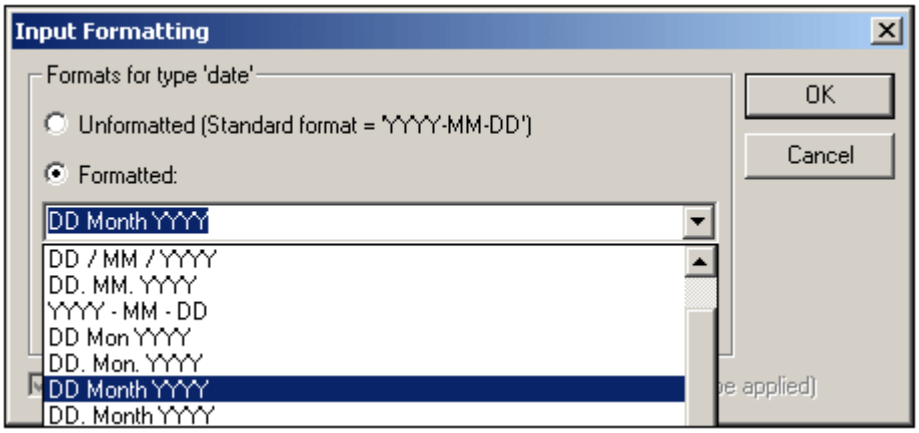
XML input fields
XML input fields (or data-entry devices) are used to enter data into the XML document via Authentic View. These input fields are text boxes, combo boxes, check boxes, radio buttons, and buttons. They are useful for constraining the data that is entered into the XML file via Authentic View. In the HTML output/s, data-entry devices have presentational value only; they cannot be used to enter data. For a complete list and description of XML input fields, see Data-entry devices.

Layout and display formatting
The XSLT processor creates an area tree which follows a model to which CSSproperties can be applied. In StyleVision, these formatting properties are available in the Text Style and Block Style windows, and they are converted to the appropriate properties for the HTML output. See Formatting for usage.

Data (or input) formatting
For nodes with certain datatypes, such as numbers and the XML Schema date datatype, the

 © 2005 Altova GmbH

node content can be formatted as required. For example, an element that contains currency amounts can be formatted so that it has two decimal places, commas after the million and thousand places, and a dollar sign before the amount (`$450,000.00`).



Such formatting, known as **Input Formatting**, appears in the Authentic View output and, for a subset of the formattable datatypes, optionally in the HTML, output.

1.7 About this documentation

Altova StyleVision 2005 documentation
The Altova StyleVision 2005 documentation consists of the following sections:

- An Introduction, which shows the relationship between StyleVision Power Stylesheets and Authentic View; describes the benefits of using Authentic View; and explains the concepts behind the working of StyleVision Power Stylesheets and Authentic View.
- A description of how to set up StyleVision and an explanation of the StyleVision interface: Getting Started.
- A tutorial to get you started using StyleVision.
- A How To Use section for commonly used StyleVision features. The How to Use section also contains a section on how to use the editing features of Authentic View. These features can be tried out in the Authentic Editor View of StyleVision.
- A section with tips on how to use StyleVision efficiently and correctly: StyleVision Tips.
- A Troubleshooting section for commonly encountered problems.
- A description of how to convert HTML to XML.
- A user reference containing a description of the various menu commands: StyleVision User Reference.

This documentation is available in the following formats:

- Onscreen Help within the application (click the **Help** menu in Altova StyleVision 2005 to access); indexed and provides text searching
- Online Help at the Support Center at the Altova Website
- Downloadable PDF, available at Documentation Download at the Altova Website
- Printed User Manual, which you can buy at Documentation Download at the Altova Website.

© 2005 Altova GmbH

Chapter 2

Getting Started

2 Getting Started

This section discusses issues related to starting work with StyleVision. It covers the following:

- A discussion about installing StyleVision and setting up related components.
- A high-level description of the StyleVision interface.
- A discussion of the document design process.

© 2005 Altova GmbH

2.1 Setting up StyleVision

In order to preview HTML output in StyleVision, you need an XSLT processor.

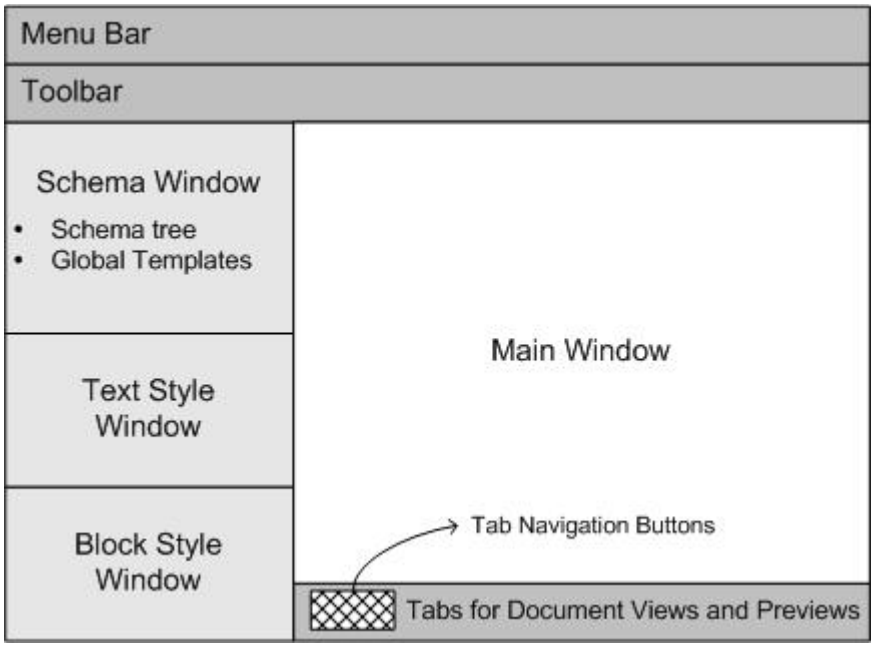
XSLT processor
StyleVision has two internal XSLT processors: the Altova XSLT 1.0 processor and the Altova XSLT 2.0 processor. No additional installation or setting is required. When you click the HTML Preview tab in the Main Window (see StyleVision interface), the XSLT processor carries out the XSLT transformation using an XSLT stylesheet generated dynamically from the StyleVision Power Stylesheet displayed in the Design tab. Which of the two processors is used is automatically determined by what XSLT version (XSLT 1.0 or XSLT 2.0) has been selected for the StyleVision Power Stylesheet. The HTML preview of the Working XML File is then displayed in the HTML Preview tab.

2.2 StyleVision interface

The graphical user interface of StyleVision is made up of two broad parts:

- The tabbed Main Window, which displays the design of the document, code of the XSLT stylesheet/s, and the output preview/s.
- The Schema Window and styling windows, in which content is selected in terms of schema elements and attributes (Schema Window), and in which items selected in the Design Document can be formatted (Text Style Window and Block Style Window).

The functions of individual parts of the interface are described below.



Title Bar
The Title Bar contains the name of the StyleVision Power Stylesheet file and the Working XML File.

Menu Bar
The Menu Bar contains the StyleVision menus.

Toolbar
The Toolbar contains icons for common commands. Some icons will be familiar from other Windows applications. Others are for specialized StyleVision actions. Note that the bottom row of the toolbar contains icons for the creation of static SPS tables and the editing of both static and dynamic SPS tables; these icons correspond to commands in the Table menu.

Main Window
The Main Window has four tabs: Design; Authentic Editor View; XSLT Stylesheet; and HTML Preview. The StyleVision Power Stylesheet is designed in the Design Window (or tab). In the Design Window, dynamic components (content that comes from the XML document) are displayed in terms of schema elements and attributes which are dragged from the schema tree; static components (such as images, non-XML text, etc) are entered directly into the Design Window. All components of the Design Document are displayed with the formatting and layout properties applied.

 © 2005 Altova GmbH

The **Authentic Editor View** tab enables you to preview the Authentic View of the Working XML File and to edit the Working XML File. In this documentation, the term **Authentic View** is used to describe the view of the XML document seen in the Authentic Editor tab. Dynamic changes that result from editing XML content are displayed in the Authentic Editor View. Changes you make to XML content in the Authentic Editor View are saved to the Working XML File.

The **XSLT Stylesheet** view shows the text of the XSLT for HTML. You are not allowed to edit the code displayed in the XSLT Stylesheet view.

Clicking the **HTML Preview** tab starts the transformation process, and displays a preview of the output. In the HTML preview, data-entry devices for XML data are displayed; however, the entry of XML data in the HTML Preview is not allowed.

The **Tab Navigation Buttons** enable you to bring tabs into view should the window width not be sufficient to accommodate all tabs. The horizontal scroll bar for the Main Window scrolls the window for the tab that is active. The Tab Navigation Buttons are, therefore, the only way to bring tabs into view without broadening the window or changing screen resolution.

Schema Window
The Schema Window displays the following:

- A **tree representation of the schema** (DTD or XML Schema) in terms of its elements and attributes. Elements and attributes are dragged from the Schema Window and placed at the required location in the Design Document.
- A list of all elements and attributes for which **global templates** are possible. When you select an element or attribute from this list, a global template is created. The formatting that you specify in a global template is applied to all occurrences of that element in the document.

Text Style Window
The Text Style Window has tabs, each with different groups of text formatting properties (such as font-weight and font-style). When text or a node containing text is selected in the Design Document, text formatting properties are applied **to the text** via the properties in the Text Style Window. Some text style properties (such as font-weight) are also available as icons in the toolbar.

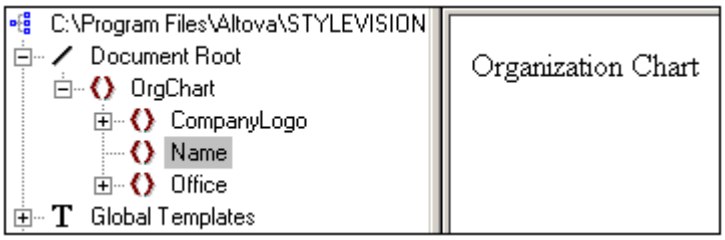
Block Style Window
The Block Style Window has tabs, each with different groups of block formatting properties (such as spacing before and after a block). When a **block component** is selected in the Design Document, block formatting properties are applied **to the block** via the properties in the Block Style Window. If the selected Design Document component is not a block, the Block Style Window is disabled (grayed out).

Note: Only one StyleVision Power Stylesheet can be open at a time in StyleVision. The StyleVision title bar and the StyleVision Power Stylesheet title bar are one and the same. Clicking the Close button in the title bar therefore closes both the StyleVision Power Stylesheet as well as StyleVision.

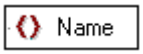
2.2.1 Schema Window

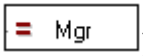
The Schema Window (shown below) displays the following:

- A **schema tree**, which is a representation of the underlying schema (DTD or XML Schema) as a tree structure.
- A list of **Global Templates** as tree structure.



Schema Tree
When the schema tree is selected, the Root Template (as opposed to Global Templates) is displayed in the Design Window, and you can create and edit the Root Template. The hierarchy of the schema tree is as follows:

- At the topmost level is the Document Root.
- At the next level is the Document Element (or Root Element); in the screenshot above, the Document element is `OrgChart`. If the underlying schema is an XML Schema (not a DTD) having more than one global element, then, in order for one of these global elements to be selected as the Document Element, you have to associate a Working XML file with the StyleVision Power Stylesheet. The document element of the Working XML File becomes the Document Element of the schema tree.
- The Document Element contains the rest of the document hierarchy. In the screenshot above, `CompanyLogo`, `Name`, and `Office` are the elements at the next level.
- The schema tree contains elements (indicated by 

) and attributes (shown with 

).

Global Templates
If the underlying schema is a DTD, all elements declared in the DTD are available in the Global Templates list as a tree structure. If the underlying schema is an XML Schema, only global elements appear in the Global Templates list.

Note: An empty global template is automatically created when you click an item in the Global Templates list. To remove a global template, right-click the item in the Global Templates list.

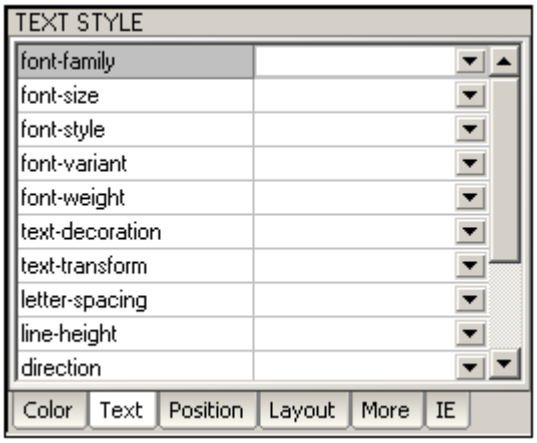
Navigating in the Schema Window
To navigate in the Schema Window, note the following:

- Items in tree views can be expanded or collapsed by clicking on the + and – icons near the element name.
- When any element or attribute in the schema is selected that element or attribute is highlighted in the Design Window.
- Clicking on a global template, causes that template to be displayed in the Design Window.
- To return to the Root Template from a global template, click either the Document Root or any element/attribute in the schema tree.

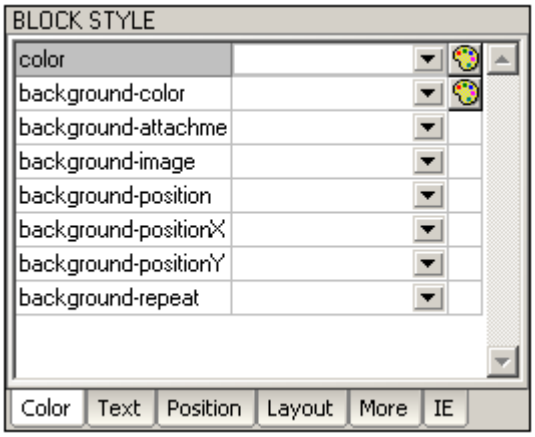
© 2005 Altova GmbH

2.2.2 Text and Block Style Windows

Text Style Window
The Text Style window allows you to apply formatting to text and to text components (such as the `(contents)` placeholder and data-entry devices). To apply text formatting, select the text or text component to be formatted and enter a value for the required property.



Block Style Window
The Block Style window allows you to apply formatting to block components. Block components include paragraphs, headings, tables, lists, etc. Any component that is not a block component (i.e. it is an inline component) can be made into a block component by assigning it a pre-defined format. When a block component is selected, the Block Style window is enabled; otherwise it is grayed out.



Note:

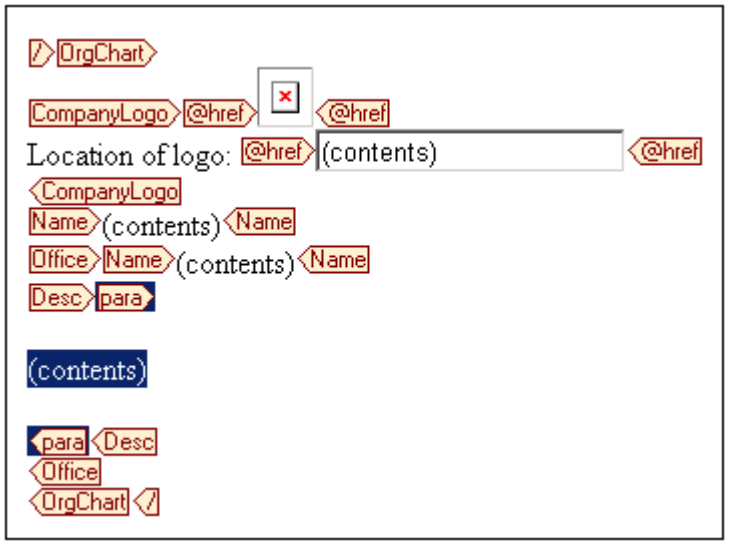
- When a property is assigned in the Block Style window, it applies to the entire block. For example, if a table is selected, and a text property is assigned in the Block Style window, then that text property applies to text in all the cells of the table. If you wish to have some property apply to the text of one particular cell only, you must select that text and assign the required property in the Text Style window.
- The Block Style Window is also useful to specify formatting properties of table cells that are not available in the Table Properties dialog. Place the cursor in the required cell, and specify a value for the required property in the Block Style Window.

Also see: Document design process and Formatting

2.2.3 Document Design Tab

In the Document Design Tab, you can design the following:

- The Root Template of the StyleVision Power Stylesheet (shown below)
- Global Templates



All aspects of the StyleVision Power Stylesheet must be designed in the Document Design window. You can edit both the structure and the formatting of your stylesheet in this view. Right-clicking on a component will allow you to edit the properties and formatting of the component. Note that you can also use drag-and-drop features as well as cut/copy-and-paste. For a complete description of the symbols and icons that appear in this tab, see Symbols and Icons.

© 2005 Altova GmbH

2.2.4 Authentic Editor View

The Authentic Editor View tab enables you to see how your Authentic XML document will look and to test it. This is particularly useful if you wish to test the dynamic features of Authentic View. For example, you could test how Authentic View behaves when you:

- Add new elements, attributes, entities
- Add new paragraphs or table rows
- Change values that affect conditional templates
- Add data via data-entry devices, for example, combo boxes, text fields, and check boxes
- Change image or link addresses

Authentic Editor View and HTML Preview
Authentic Editor View differs from HTML in that Authentic Editor View enables data-entry to the XML document, thus enabling dynamic changes to be made. StyleVision creates a temporary XML file based on the Working XML File, which is what is displayed in Authentic Editor View. This temporary XML file accepts data from Auhentic Preview, and any modification you make affects the display of the XML document in Authentic Editor View.

The HTML Preview, on the other hand is exclusively an output document, produced by a one-time transformation of the XML document. You cannot enter data in the HTML Preview. The data that is being acted upon is that in the Working XML File. Any change you make in the Document Design acts on the data in the Working XML File, and a new transformation is carried out each time you switch to the HTML Preview.

Authentic Editor View Limitations
Authentic Editor View is similar to the full-fledged Authentic View available in XMLSpy 2005 and Authentic 2005 Desktop Edition except in the following major respects:

- Entry Helpers are not available in the interface. To insert or append nodes, you must right-click and use the context menus.
- You cannot insert XML tables.

A full description of how to use Authentic View features is given in the How to Use section. For additional reference, please see the Authentic View tutorial in the XMLSpy 2005 or Authentic 2005 Desktop Edition documentation.

2.2.5 XSLT-HTML Tab

The XSLT-HTML tab displays the XSLT stylesheet for transformation to HTML. This view is useful for debugging the XSLT stylesheet and for planning how to process the generated XSLT. It also enables you to see the code of the HTML document that will be created—which can be useful for designing HTML-related files, such as CSS and JavaScript.

Note the following points about the XSLT-HTML tab:

- Any change made in the Design Document is immediately reflected in the XSLT-HTML tab. To view the change in the XSLT stylesheet, switch from the Design Document to the XSLT Tab.
- No editing is allowed in this view. All editing is done in the Document Design tab.
- The XSLT-HTML tab features line-numbering and expandable/collapsible elements. To expand or collapse an element, click the + and – icons in the left margin.
- Text searches are enabled (select **Edit | Find**).

© 2005 Altova GmbH

2.2.6 HTML Preview

The HTML Preview shows an exact preview of the HTML file that will be obtained by transforming the Working XML File using the XSLT-for-HTML that you generate for the current StyleVision Power Stylesheet (using **File | Save Generated Files | Save Generated XSLT-HTML File...**). You can use the generated XSLT file to transform any other XML file based on the same schema as that on which the StyleVision Power Stylesheet is based.

If you make a change in the Design Document, you can view the effect of this change on the HTML output by switching to the HTML Preview. Note, however, that although the HTML Preview may look identical to the Authentic Editor View, it is different in one important respect. Unlike the Authentic Editor View, it does not accept XML data input. So components which are intended for XML data input, such as check boxes, combo boxes, etc, will not accept any input in HTML Preview.

Note: HTML Preview is enabled only if a Working XML File has been assigned to the StyleVision Power Stylesheet. If none is assigned, you will be prompted to assign a Working XML File when you click HTML Preview.

2.2.7 Menu Bar and Toolbar

Menu Bar
The menu bar contains the various application menus. The following conventions apply:

- If commands in a menu are **not** applicable in a view or at a particular location in the document, they are grayed out and disabled.
- Some menu commands pop up a submenu with a list of additional options. Menu commands with submenus are indicated with a right-pointing arrowhead to the right of the command name.
- Some menu commands pop up a dialog that prompts you for further information required to carry out the selected command. Such commands are indicated with an ellipsis (...) after the name of the command.
- To access a menu command, click the menu name and then the command. If a submenu is indicated for a menu item, the submenu opens when you mouseover the menu item. Click the required sub-menu item.
- A menu can be opened from the keyboard by pressing the appropriate key combination. The key combination for each menu is **Alt+KEY**, where KEY is the underlined letter in the menu name. For example, the key combination for the **File** menu is **Alt+F**.
- A menu command (that is, a command in a menu) can be selected by sequentially selecting (i) the menu with its key combination (see previous point), and then (ii) the key combination for the specific command (**Alt+KEY**, where KEY is the underlined letter in the command name). For example, to create a new file (**File | New**), press **Alt+F** and then **Alt+N**.
- Some menu commands can be selected **directly** by pressing a special **shortcut** key or key combination (**Ctrl+KEY**). Commands which have shortcuts associated with them are indicated with the shortcut key or key combination listed to the right of the command. For example, you can use the shortcut key combination **Ctrl+N** to create a new file; the shortcut key **F8** to validate an XML file.

Toolbar
The toolbar contains buttons that are shortcuts for commands found in the menus. The name of the command appears when you place your mouse pointer over the button. To execute the command, click the button.

Toolbar buttons are arranged in groups. In the **Tools | Customize | Toolbars** dialog, you can specify which toolbar groups are to be displayed. In the GUI, you can also drag toolbar groups to alternative locations by clicking and dragging a toolbar handle to the desired location.

© 2005 Altova GmbH

2.3 Working with StyleVision

The Altova StyleVision 2005 icon  will appear in your **Start | Programs** menu after the application has been installed. In working with StyleVision your broad objectives would be as follows:

- Create a StyleVision Power Stylesheet. The StyleVision Power Stylesheet is required in order to be able to edit XML documents in Authentic View. The StyleVision Power Stylesheet is also the source from which XSLT stylesheets are generated.
- Generate and save XSLT stylesheets and their output files.
- Convert an HTML file to an XML file.

In this section, we outline the process used to design a StyleVision Power Stylesheet (*see* Document design process) and explain how XSLT stylesheets and their output files are saved. Using StyleVision from the command line is described in StyleVision from the command line, and HTML-to-XML conversions are explained in the section HTML to XML Conversion.

2.3.1 Document design process

The document you create in StyleVision is a StyleVision Power Stylesheet, and it is created in the Design tab of the Main Window. The process for creating a StyleVision Power Stylesheet (known also as the Design Document) can be divided into the following general parts:

Selection of dynamic content and its placement in document

- **Schema:** A schema (DTD or XML Schema) is used as the starting point. The schema, which provides the structure (in terms of elements and attributes) according to which an XML document can be built, is loaded into StyleVision. It is displayed as a **tree** in the Schema Window. Your StyleVision Power Stylesheet is based on the schema you load into StyleVision. Be sure to finalize your schema before starting to build the StyleVision Power Stylesheet. If you change your schema subsequently, you might need to re-create your StyleVision Power Stylesheet from scratch.
- **XSLT version:** After loading the schema, select the XSLT version you wish to use by clicking the XSLT 1.0 or XSLT 2.0 toolbar icons.
- **Content to be selected:** XML content is selected by dragging an element or attribute from the schema tree.
- **Location in output document:** The dragged element/attribute is placed in the **design document**. Where it is dropped determines the **location** of that particular element/attribute in the design document. Since the actual content (i.e. data) comes from the XML document and could vary depending on the XML content of the element/attribute, this kind of content is known as dynamic content and its containing component is known as a dynamic component.

StyleVision Power Stylesheet Components

StyleVision Power Stylesheet components can be categorized in terms of:

- Type (static or dynamic), and
- Form (e.g. contents, data-entry device, table, image, etc).

Type of component
Static components are those with static content, that is, content entered in the StyleVision Power Stylesheet; static content does not come from the XML document. The most common examples of static content are text and images. Static text is text that is entered directly in the StyleVision Power Stylesheet. Static images are those that have their URLs specified in the StyleVision Power Stylesheet. Dynamic components, on the other hand, are those components which have content that comes from the XML document.

Form of component
This refers to the form the component takes, which could be text, or a table, image, hyperlink, etc. Two forms that require explanation are the `(contents)` placeholder and data-entry devices. The `(contents)` placeholder is used with dynamic components, specifically an element or attribute, and serves to output the contents of the node. Data-entry devices are described in the section Data-entry devices.

Inserting static components

- **Static components:** Additional components that are external to the XML source can be added to the design document. These components could be text, images, table structures, hyperlinks, etc. They are considered static because they are non-XML. Since they are sourced in the stylesheet, they cannot be changed in the XML document; they are inserted in the output document from the stylesheet at the time of transformation.

Processing of dynamic content

 © 2005 Altova GmbH

- **Basic processing:** When an element/attribute is dropped into the design document, you must decide in what form its content is to be displayed. Some possibilities are as follows. If the content model of the selected node is text, you might want to create the content as a paragraph. If the content model of the selected node is other elements, you might want to create the selected element as a table, with the child elements forming the columns of the table. If the selected node is an attribute, you might want to create the attribute as an image, with the value of the attribute being the location of the image. Or yet again, you may wish to create the selected node as a data-entry device to accept XML input in Authentic View.
- **Advanced processing:** You can build conditional templates to process a node in alternative ways depending on the value of any node in the XML document. You can also use XPath expressions (i) to generate XML values from other XML content, and (ii) to perform additional validation of nodes.

Formatting of content

- Precise formatting can be applied to any static or dynamic component in the design document. Formatting properties include text and block styling, as well as those for complex table layouts.

The above provides a broad outline of how a StyleVision Power Stylesheet is designed. The more detailed aspects are explained in subsequent sections.

2.3.2 Saving generated files

When you save the document in the Design tab, it is saved as a StyleVision Power Stylesheet (`.sps` file). In order to generate and save XSLT stylesheets based on your design, you must explicitly generate and save these files. Additionally, you can process the Working XML File with a stylesheet and save the resultant output.

You can save these generated files using one of two methods:

- Using the File | Save Generated Files command
- Calling StyleVision from the command line

There is one significant difference between the two methods. If you use the **File | Save Generated Files** command, files assigned to the StyleVision Power Stylesheet are used for the processing, for example, the Working XML File.

Note:

- You must save all changes to the StyleVision Power Stylesheet before saving generating files.
- Output HTML files are generated, by default, by processing the Working XML File assigned to the StyleVision Power Stylesheet. If you wish to use an alternative XML file, you can use StyleVision from the command line.

© 2005 Altova GmbH

Chapter 3

Tutorial

3 Tutorial

The aim of this tutorial is to create a StyleVision Power Stylesheet for a company with offices in the US and EU. After the StyleVision Power Stylesheet has been created, it can be used to edit an XML document in Authentic View. To do this, the XML document must be linked to the StyleVision Power Stylesheet. The following general points apply to this tutorial.

Previewing the outputs and checking for differences among them
While you create the StyleVision Power Stylesheet in StyleVision, you can directly preview the StyleVision Power Stylesheet and HTML output in StyleVision. To do this you must assign a Working XML File to the StyleVision Power Stylesheet.

You should also test the Authentic View of the Working XML File by editing the file in Authentic Editor View. The Authentic Editor View is different from the HTML Preview in that the Working XML File can be edited in Authentic Editor View but not in HTML Preview. As a result, dynamic functionality, such as Auto-Calculations and Conditional Template processing can be tested in Authentic Editor View but not in HTML Preview.

Remember that the previews you see in StyleVision are of the Working XML File you have assigned to your StyleVision Power Stylesheet. If you modify and re-save your Working XML File outside StyleVision subsequent to opening a StyleVision Power Stylesheet that uses the Working XML File, then you must close and reopen your StyleVision Power Stylesheet in order to see the changes in the revised XML document. Note that you can also save changes to the Working XML File from Authentic Editor View (see Save Authentic XML Data).

Text View of Working XML File
Occasionally, you might want to see the text of your Working XML File to check whether changes made in Authentic View have been applied to the XML. You can see the text by viewing the Working XML File in the Text View of your XML editor, or by opening the file in a text editor.

Tutorial files
The following files are used in this tutorial. They are in the `Examples` folder of your application:

- StyleVision Power Stylesheet: `NanonullOrg.sps`
- XML Schemas: `NanonullOrg.xsd` and the imported schema `address.xsd`.
- Working XML File: `NanonullOrg.xml`. Assign this file to the StyleVision Power Stylesheet as the Working XML File.

You will create your StyleVision Power Stylesheet from scratch. When you have completed it, it should be similar to the finished StyleVision Power Stylesheet `NanonullOrg.sps`. Before you start work on the tutorial, rename the supplied `NanonullOrg.sps` to something else, such as `NanonullOrg_original.sps`. It is also a good idea to make a copy of the XML file `NanonullOrg.xml` before you start editing it.

Note: StyleVision supports **unlimited undo**. So you can go back as many steps as you want even after saving the file. Keep this in mind while doing the tutorial; it could be useful.

© 2005 Altova GmbH

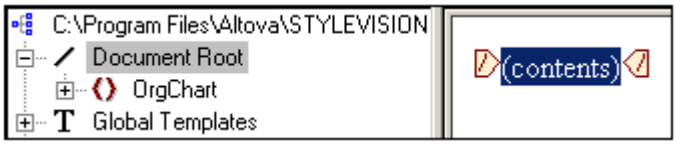
3.1 Setting up the StyleVision Power Stylesheet

Setting up a StyleVision Power Stylesheet entails the following:

- Opening a schema in StyleVision
- Selecting the XSLT version for the StyleVision Power Stylesheet
- Assigning a Working XML File
- Creating a rudimentary document outline
- Saving the StyleVision Power Stylesheet with a name

Starting StyleVision and opening a schema

1. Start StyleVision by double clicking the StyleVision icon on your Desktop or by clicking **Start | All Programs | Altova StyleVision 2005**. StyleVision starts. The Schema Window and the Design Document window are empty.
2. Select the menu option **File | New** and open the supplied `NanonullOrg.xsd` schema file (in the `Examples` folder of your Altova StyleVision 2005 installation folder).



This creates a schema tree with the element `OrgChart` as the root element in the Schema Window, as well as entries for Global Templates immediately below the schema tree. A `(contents)` placeholder for the root element is also created in the Design Document window.

Note: We use the word schema (not capitalized) to denote the superset that contains DTDs and XML Schemas (capitalized). While this tutorial uses an XML Schema (`NanonullOrg.xsd`), note that you can also use DTDs in StyleVision.

Selecting the XSLT version for the StyleVision Power Stylesheet
We want this StyleVision Power Stylesheet to conform to XSLT 1.0. So in the toolbar, click the XSLT 1.0 icon.

Encoding of output file
In the Encoding tab of the Options dialog (**Tools | Options...**), ensure that HTML encoding is set to UTF-8.

Assigning the Working XML File
The Working XML File is used to preview how an XML file based on the selected schema will be processed by your StyleVision Power Stylesheet. StyleVision provides an Authentic Editor View and a preview of the HTML output of the Working XML File so you can quickly see how your StyleVision Power Stylesheet will process the XML.

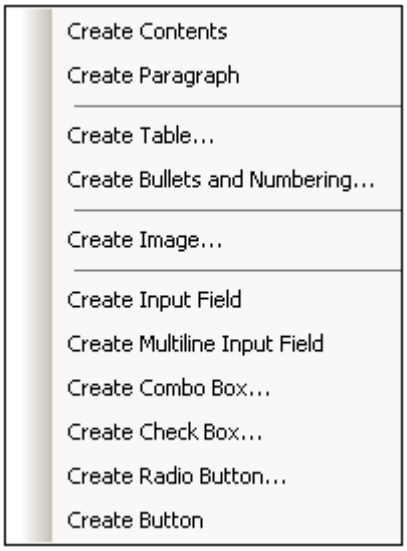
To assign a Working XML File, select **File | Assign Working XML File...**. Then browse and select `NanonullOrg.xml`, which is in the `Examples` folder. Before assigning the Working XML File, you should save a copy of the original `NanonullOrg.xml` with some other name.

The Working XML File, including the path to it, now appears in the title bar.

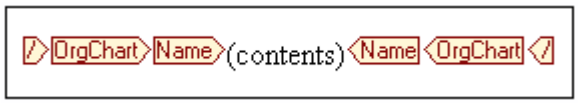
Creating the document outline in the Design view

1. In the Schema Window, expand the `OrgChart` element.
2. Select the `/OrgChart/Name` element and drag it into the Design window. Position it

just after the `(contents)` placeholder. A blinking cursor appears after the `(contents)` placeholder, and the dragged element becomes an arrow. When you release the mouse button, a popup menu appears:



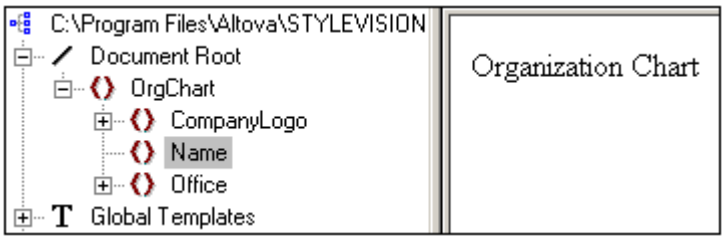
3. Click the **Create Contents** option in the popup. The Design Document now looks like this:



The `Name` element with a `(contents)` placeholder is inserted. The `(contents)` placeholder represents the content of the `Name` element. When the XML file is processed with this StyleVision Power Stylesheet (for Authentic View and HTML output), the XML content of the `Name` element will be displayed.

Previewing the XML file
To preview the Authentic View and HTML outputs, we will use the Authentic Editor View of StyleVision.

- Authentic Editor View (shown below) provides an accurate preview of both the Authentic View and HTML outputs. The HTML Preview looks the same.



If you look in the Text View of your Working XML File (in an XML editor or text editor), you will see that the contents of the `/OrgChart/Name` element is "Organization Chart", which is what appears in the Authentic Editor View. Also click the HTML Preview tab to see a browser preview of the HTML.

Viewing the XSLT stylesheets
To view the XSLT stylesheet (for HTML output), click the XSLT Stylesheet tab at the bottom of the Main Window. It will look something like this (syntax coloring has been added and

 © 2005 Altova GmbH

indentation changed in the code below so as to aid readability):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:n1="http://www.xmlspy.com/schemas/orgchart"
  xmlns:ipo="http://www.altova.com/IPO"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <xsl:template match="/">
    <html>
      <head />
      <body>
        <xsl:for-each select="n1:OrgChart">
          <xsl:for-each select="n1:Name">
            <xsl:apply-templates />
          </xsl:for-each>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

You are not allowed to edit the XSLT code. If you wish to modify these XSLT files, you can save them as a generated XSLT file, and then edit the generated files in XMLSpy 2005 or other editor. If you do this, however, note that the modified file will be different from the StyleVision Power Stylesheet, which you can modify only in the Design view of StyleVision.

A generated XSLT stylesheet can be saved by selecting the menu command **File | Save Generated Files**, and then selecting the stylesheet from the submenu.

Saving the StyleVision Power Stylesheet

The StyleVision Power Stylesheet is saved as an SPS (`.sps`) file; it controls the display and processing of the Authentic View of an XML document. The SPS file also contains information about related files, such as the underlying schema, Working XML File, and Template XML File. (See StyleVision concepts and terminology for a description of these terms.)

To save the StyleVision Power Stylesheet, click **File | Save Design**, and enter `NanonullOrg.sps` as the name of the file. (Be sure to save a copy of the original (supplied) `NanonullOrg.sps` file under another name.) The name you enter appears in the title bar.

3.2 Processing attribute values for output

The Design Document defines how the XML document is to be processed. XML content is included as follows. First, the required XML content (element or attribute) is dragged from the schema tree. It is dropped into the Design Document as one of several types of StyleVision Power Stylesheet components (such as contents, data-entry device, table, etc). Then, further XSLT/XPath processing can be specified for it. Finally the required formatting is applied.

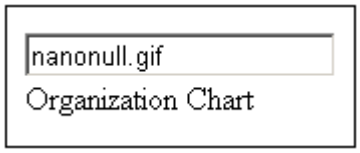
In this section you will learn how to include attribute values in your output.

Including an attribute as a data-entry device
In the schema tree, you will see that the element `CompanyLogo` has an `href` attribute, the value of which gives the location of the logo image. Now let us say we want the user of Authentic View to specify what image should be used as the logo. We can create the `href` attribute as an input field, and the user can enter the location of the image here. This is how:

1. In the Design window, place the cursor between the start tags of `OrgChart` and `Name`. Press Enter to put the `Name` element on the next line.
2. Drag the `href` attribute from the Schema Window into the Design Document. Drop it just after the `OrgChart` start tag.
3. In the pop-up menu that appears, select "Create Input Field". Your Design Document will now look like this:



The Authentic and HTML Previews will look like this (the data comes from the Working XML File and is the value of the `href` attribute):



Note that the `href` attribute has been created as an input field, while the `Name` element was created as `(contents)`. The Authentic View user will be able to edit both types of display. The input field, however, indicates more clearly to the user that data in this field can be modified.

Adding non-XML XSLT text
In order to indicate what data is contained in the input field, text can be added before the input field. This text is part of the StyleVision Power Stylesheet; it does not come from the XML document.

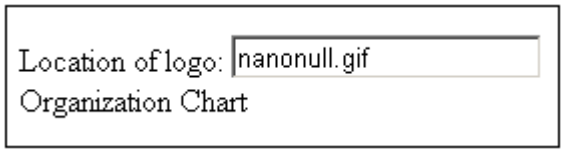
To add text to the XSLT document, type it into the Design Document at the required location. In your StyleVision Power Stylesheet, do the following:

1. Place the cursor before the start tag of the `CompanyLogo` element and press Enter. This adds a line break.
2. Type "Location of logo: " before the opening tag of the `href` attribute. The Design Document now looks like this:

© 2005 Altova GmbH



The Authentic and HTML Previews will look like this:



A note on data-entry devices
Data-entry devices are used by the **Authentic View user** to enter **XML data into the XML document**. In an HTML file that has been created with an XSLT generated from an StyleVision Power Stylesheet, a data-entry device cannot be used for data-entry purposes. Data-entry devices in HTML output **only display** the relevant XML data. No input is accepted in these formats. The data-entry device is intended for input of XML data in **Authentic View**.

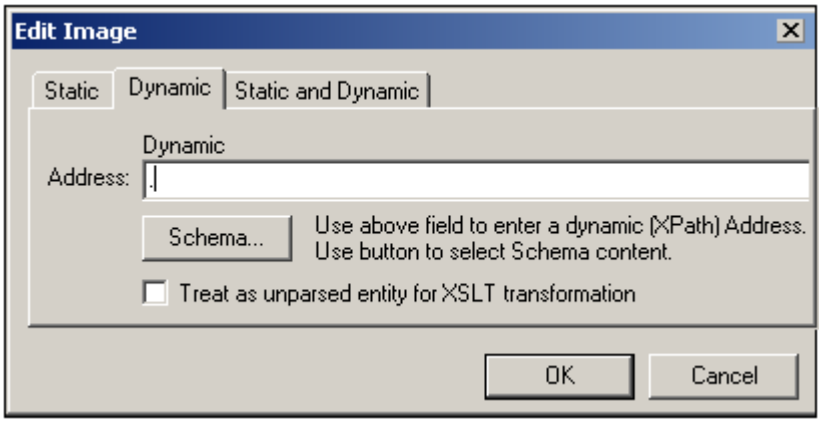
3.3 Including an image

An image can be included in the output of the XML document by including a link to it in the StyleVision Power Stylesheet. If the link is explicitly specified in the StyleVision Power Stylesheet, the included image is referred to as a static image—because its address does not, and cannot, be changed in the XML document. If, on the other hand, the StyleVision Power Stylesheet specifies, instead of the image address, a location in the XML document where the address is to be found, then the image is dynamic; it changes when the corresponding XML content changes.

Inserting a dynamic image

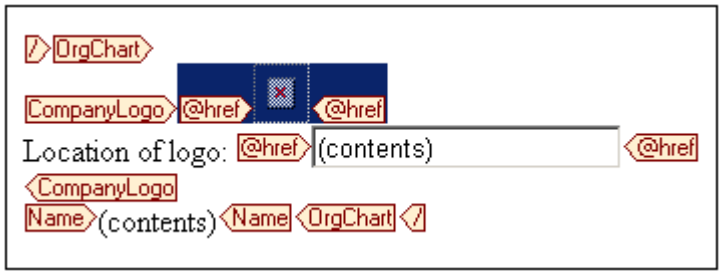
In the previous section, we created an input field for the value of the `href` attribute. This value provides the address of the logo image, and it can be specified by the Authentic View user. Now we will create a display for the image that is specified in the `href` attribute value. Do the following:

1. Drag the `href` attribute from the schema tree, and drop it immediately after the start tag of the `CompanyLogo` element.
2. In the pop-up menu that appears, select Create Image.
3. In the Insert Image dialog (shown below), switch to the Dynamic tab.



4. Check that the Address field contains a period. This is the XPath expression for the current node, which is the `href` attribute (this is the node that was dragged from the schema tree). If there is no period, then type in one.
5. Click OK.
6. Save your changes.

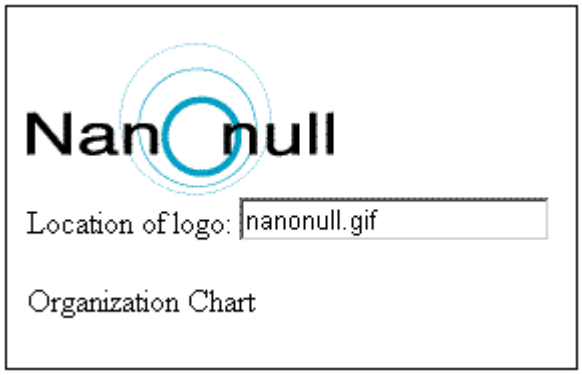
The Design Document will now look like this:



Notice that the `href` attribute appears twice in the StyleVision Power Stylesheet: first as an image, then as an input field. This way the user can specify in the input field what image is to be used; the specified image will be displayed immediately at the point where the image has been inserted.

© 2005 Altova GmbH

The Authentic and HTML Previews will look like this:



Now, in Authentic Editor View, change the value of the `href` attribute and see the dynamic effect on the logo display. Do it this way:

1. In Authentic Editor View, place your cursor in the "Location of logo" input field.
2. Replace `nanonull.gif` with `altova.gif` (or any other image in any folder); use a relative path to specify the logo's location. Your Authentic Editor View will look like this.



3. Now change the logo back to `nanonull.gif`.

Note: You cannot change XML values in the HTML Preview of StyleVision since this is **an output** of the XML document. Authentic View, on the other hand, is **a view** of the XML document.

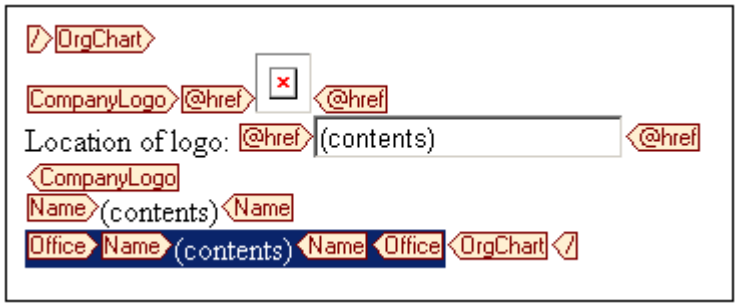
3.4 Processing element content for output

The processing of element content is defined in the Design Document in the same way as the processing of attribute values is: drag the element from the schema tree, drop it into the Design Document as a StyleVision Power Stylesheet component, specify further XSLT/XPath processing, and apply the required formatting.

In this section you will add element content to the Design Document. The objective is to create the contents of the `//Office/Name` element on one line, and the contents of the `//Desc/para` elements below it. This is how you will do it.

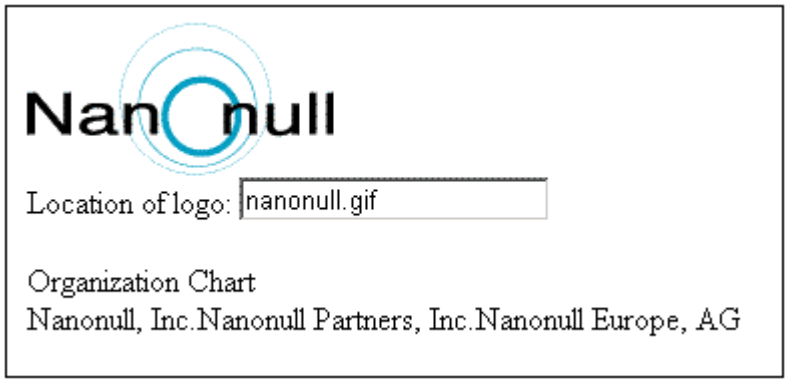
Inserting the `//Office/Name` element

1. Place the cursor before the end tag of the `OrgChart` element and press Enter to insert a line break.
2. Expand the `Office` element in the schema tree.
3. Drag the `//Office/Name` element to the location before the `OrgChart` end tag, and create the element as contents. Your Design Document will look like this.



Notice that tags are created for all levels of the hierarchy between the `//Office/Name` element and the containing `OrgChart` element.

The Authentic and HTML Previews will look like this:

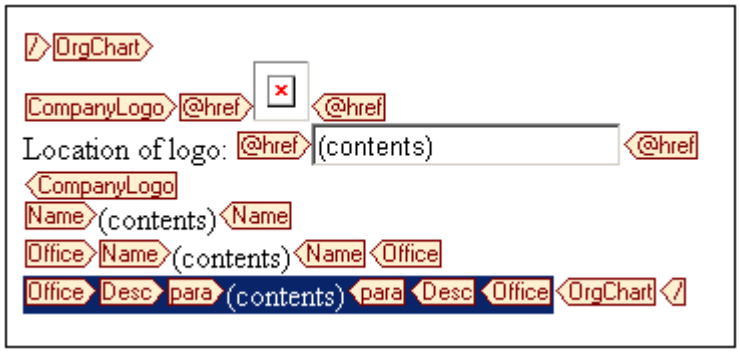


The Working XML File has three `//Office/Name` elements. Notice that the contents of all three are output (without any whitespace between elements).

Create the `//Office/Desc/para` element

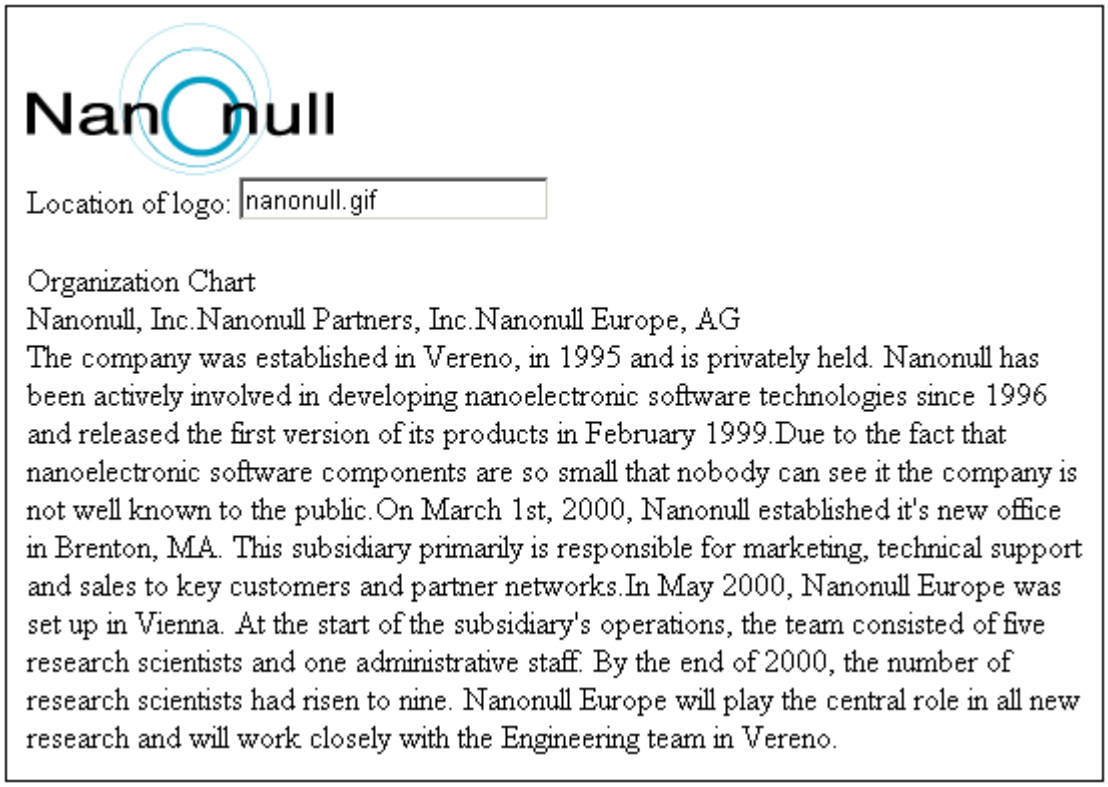
1. Put a line break before the `OrgChart` end tag.
2. Drag the `//Office/Desc/para` element to the location before the `OrgChart` end tag, and create the element as contents. Your Design Document will look like this.

© 2005 Altova GmbH



Notice that tags for all ancestors of `para` are created up to but excluding the containing ancestor (which is `OrgChart`).

Now look at the Authentic or HTML Preview.

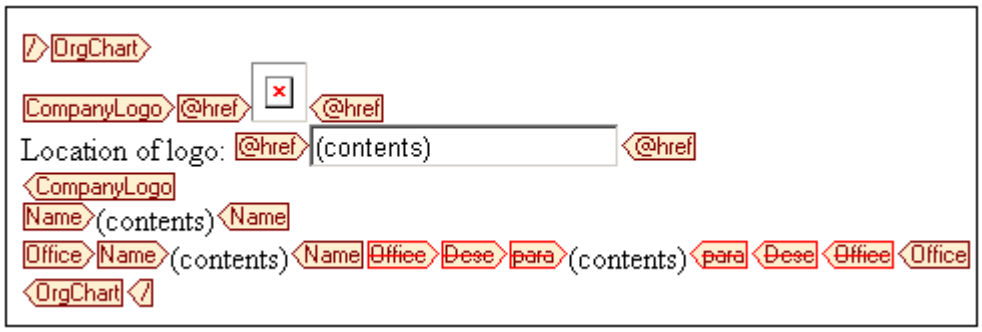

Nanonull
Location of logo:
nanonull.gif
Organization Chart
Nanonull, Inc.Nanonull Partners, Inc.Nanonull Europe, AG
The company was established in Vereno, in 1995 and is privately held. Nanonull has been actively involved in developing nanoelectronic software technologies since 1996 and released the first version of its products in February 1999.Due to the fact that nanoelectronic software components are so small that nobody can see it the company is not well known to the public.On March 1st, 2000, Nanonull established it's new office in Brenton, MA. This subsidiary primarily is responsible for marketing, technical support and sales to key customers and partner networks.In May 2000, Nanonull Europe was set up in Vienna. At the start of the subsidiary's operations, the team consisted of five research scientists and one administrative staff. By the end of 2000, the number of research scientists had risen to nine. Nanonull Europe will play the central role in all new research and will work closely with the Engineering team in Vereno.

All three `//Office/Name` elements are displayed before all the `//Office/Desc/para` elements, which are run together. This is not the desired result. What we want is the content of the first `//Office/Name` element followed by the content of the `Desc/para` elements for that office, and then the content of the second `//Office/Name` element followed by the contents of the `Desc/para` elements for that office; and so on.

If you look at your Design Document, you will see that all the `//Office/Name` elements are processed first, and then all the `//Office/Desc/para` elements. In order to process the `Name` child element of an `Office` element followed immediately by the `Desc/para` elements of that `Office` element, you need to change the location of the `//Office/Desc/para` element so that it is included in a single `Office` element loop and follows the `Name` element directly.

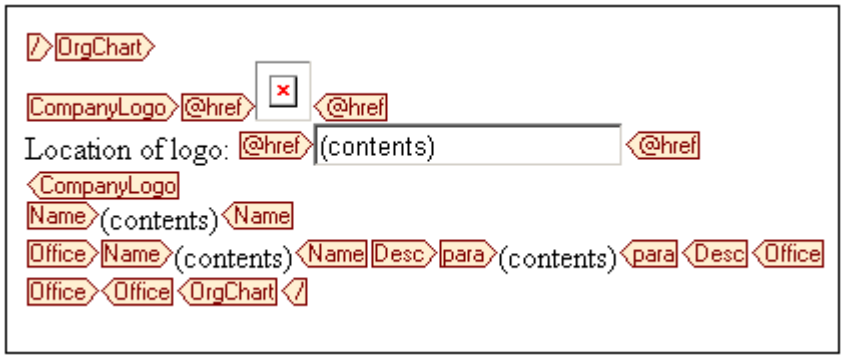
Changing the location of the `//Office/Desc/para` element

1. Select the `//Office/Desc/para` element by clicking the start tag of its `Office` element.
2. Drag the selected element and drop it before the end tag of the `Office` element on the line above. An error message pops up saying that a template selects a node that does not exist.

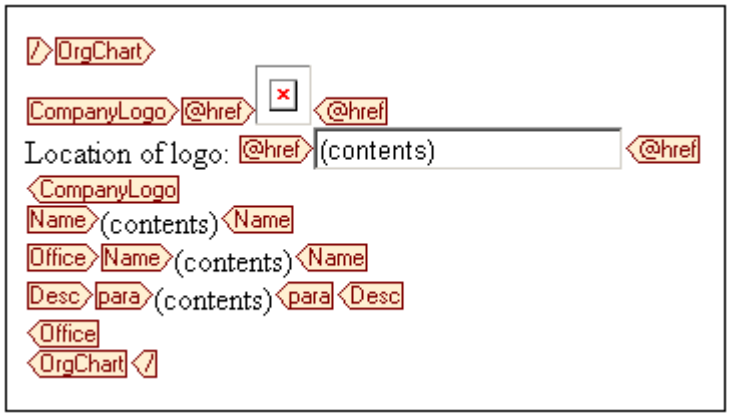

/ OrgChart
CompanyLogo @href @href
Location of logo: @href (contents) @href
CompanyLogo
Name (contents) Name
Office Name (contents) Name Office Desc para (contents) para Desc Office Office
OrgChart /

Notice that the tags of the non-existent element have a red strikethrough. The XPath `Office/Desc/para` within the containing `Office` element is wrong because the XPath in context then is: `//Office/Office/Desc/para`; that's one `Office` element too many. The correct XPath in the context of the `Office` element is `Desc/para`, not `Office/Desc/para`.

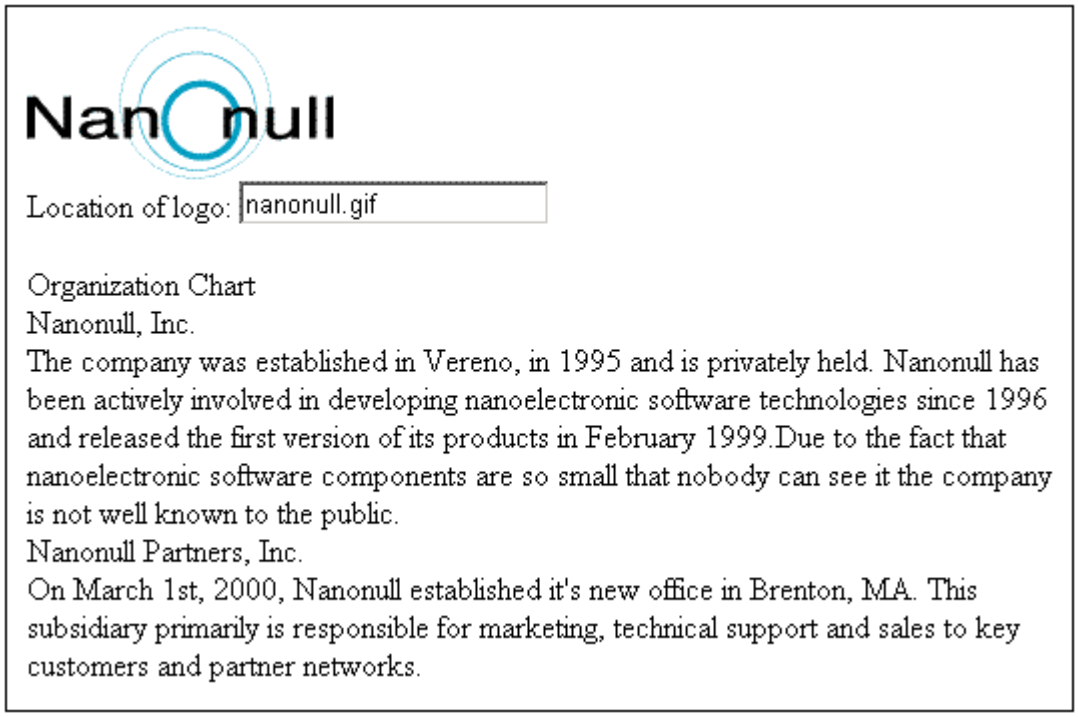
3. In the error dialog, click OK, then press Ctrl + Z to undo the move.
4. Select the `Desc/para` element by clicking the start `Desc` tag.
5. Drag the `Desc/para` element and drop it before the end tag of the `Office` element on the line above. Your Design Window will now look like this.


/ OrgChart
CompanyLogo @href @href
Location of logo: @href (contents) @href
CompanyLogo
Name (contents) Name
Office Name (contents) Name Desc para (contents) para Desc Office
Office Office OrgChart /

6. Delete the `Office` start and end tags on the last line, and put a line break after the `//Office/Name` element and after the `//Office/Desc` element. Your Design Window should now look like this.



The Authentic and HTML Previews will look like this:


Nanonull
Location of logo: nanonull.gif
Organization Chart
Nanonull, Inc.
The company was established in Vereno, in 1995 and is privately held. Nanonull has been actively involved in developing nanoelectronic software technologies since 1996 and released the first version of its products in February 1999.Due to the fact that nanoelectronic software components are so small that nobody can see it the company is not well known to the public.
Nanonull Partners, Inc.
On March 1st, 2000, Nanonull established it's new office in Brenton, MA. This subsidiary primarily is responsible for marketing, technical support and sales to key customers and partner networks.

3.5 Formatting the components

The formatting that can be applied to a StyleVision Power Stylesheet component is of two types: text formatting and block formatting. Text formatting is applied to text, and includes font sizes, font weights, and font colors. Block formatting can be applied to components that have been explicitly defined as blocks, and include background colors for the block, spacing around the block, and borders for the block. A component can be explicitly defined as a block by assigning it a pre-defined format.

To apply formatting to a component, the following three mechanisms are used, either alone or in combination:

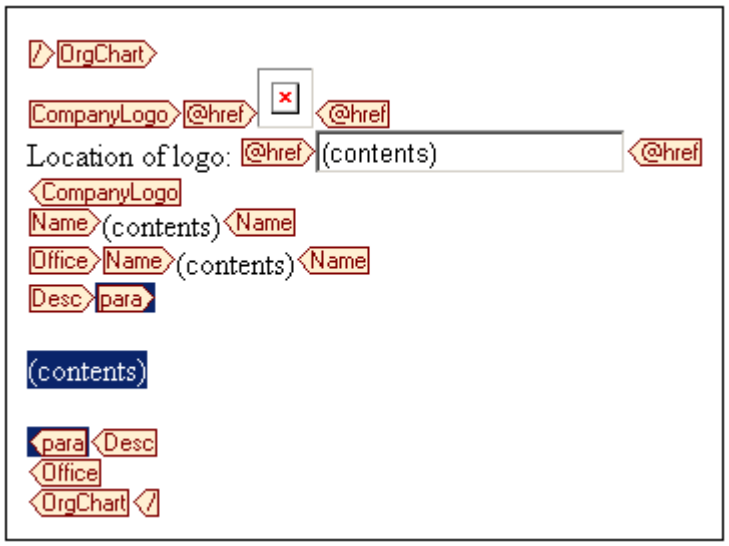
- Applying a pre-defined format (a pre-defined format includes text and block formatting, and is also used to define a component as a block)
- Applying Text Style properties (in the Text Style Window or using toolbar icons)
- Applying Block Style properties (in the Block Style Window; additional properties are available with toolbar icons)

In this section, you will learn to apply formatting using all three mechanisms.

Applying a pre-defined format to para

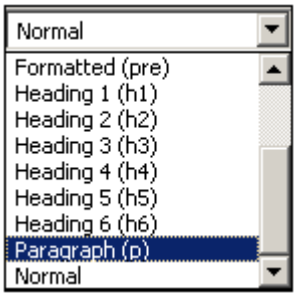
1. Select the `//Desc/para` element by clicking the start tag of the `para` element.
2. Right-click, and select **Change to | Paragraph**.

This creates the pre-defined paragraph formatting **for each** para element child of `Desc`. The Design Document now looks like this:



Alternatively, select the `(contents)` placeholder of the `//Desc/para` element (by clicking the placeholder), and in the drop-down Pre-defined Format menu in the toolbar (shown below) select **Paragraph (p)**.

© 2005 Altova GmbH

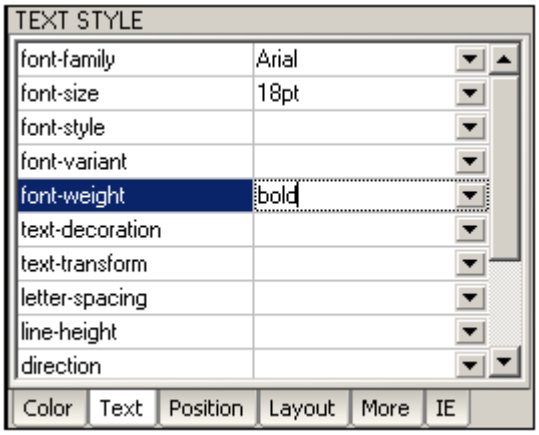


This applies the same formatting as above. Note that in the first case, the **`para` element was selected** and its processing was changed from normal to paragraph; in the second case the **`(contents)` placeholder was selected** and a pre-defined paragraph format was applied. In both cases, each `para` child element of `Desc` is created as a paragraph.

Note: If you select the `para` element—and not the `(contents)` placeholder—and then apply the pre-defined paragraph format (from the Drop-down Format menu in the toolbar), the contents of all `para` elements in that `Desc` element will run on and be displayed as a single paragraph, i.e. the entire selection is formatted as a single paragraph.

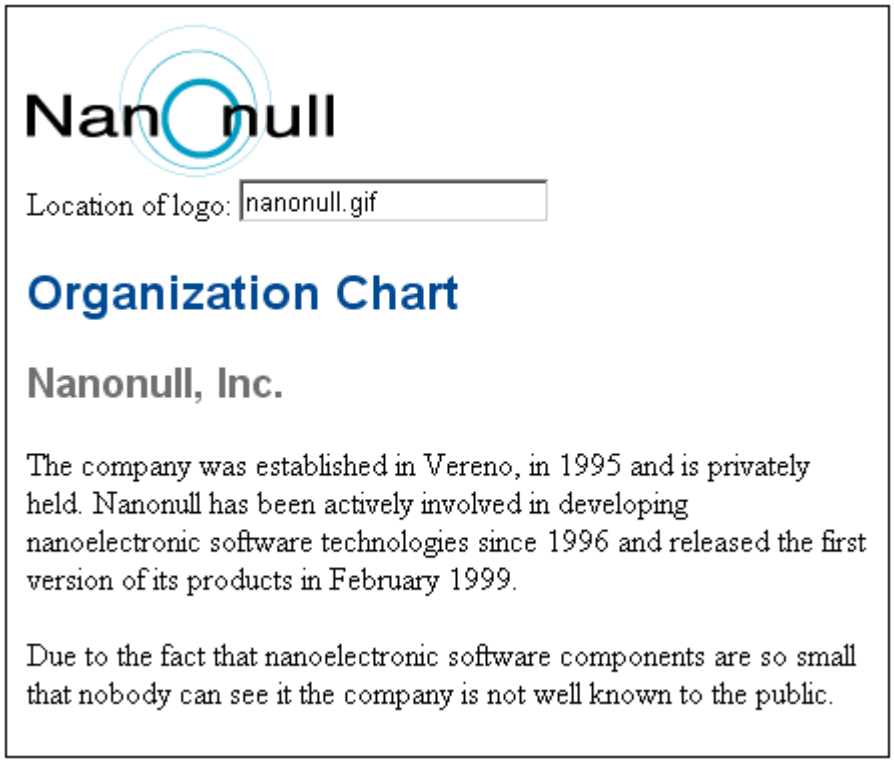
Specifying Text Style properties

1. Select the `(contents)` placeholder of the `/OrgChart/Name` element. Notice that the Text Style Window is enabled, but the Block Style Window is disabled (grayed-out). This is because the selected component is not a block. It gives the appearance of being a block because it is on a line by itself. This effect, however, was created by a line-break—which by itself **does not** make the component a block.
2. In the Text Style Window, select the Text tab, and set the font to Arial, 18pt, bold.



3. Now select the color tab, and specify a color for the text. The component in the Design Window displays the specified formatting.
4. Check the Authentic or HTML Preview, and make changes if required.
5. In the Design Window, select the `(contents)` placeholder of the `//Office/Name` element, set the font to Arial, 15pt, bold, and give it a color.
6. Insert a line break before the `Office` element.
7. Check the Previews, and take a look at the XSLT code.

Your Authentic and HTML Previews should look something like this:


Nanonull
Location of logo: nanonull.gif
Organization Chart
Nanonull, Inc.
The company was established in Vereno, in 1995 and is privately held. Nanonull has been actively involved in developing nanoelectronic software technologies since 1996 and released the first version of its products in February 1999.
Due to the fact that nanoelectronic software components are so small that nobody can see it the company is not well known to the public.

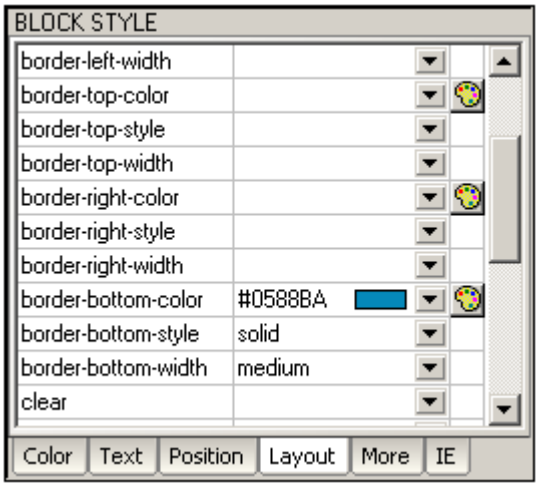
Note: To control vertical spacing for text components, use the line-break feature in the Design Window and/or the properties in the Position tab of the Text Style window.

Text formatting icons: You can also make text bold, italic, or underlined by highlighting the text and then clicking the appropriate icon in the toolbar. These icons are shortcuts to creating the required values in the Text Style Window.

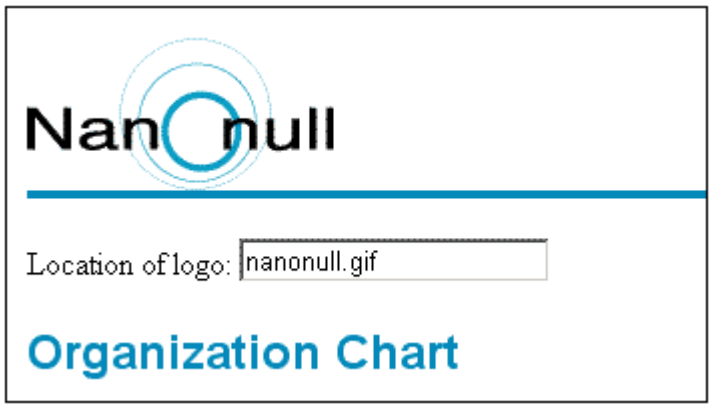
Specifying Block Style properties
Block Style properties are applied to blocks via the Block Style Window. In order for this window to be enabled, the selected component must be a block. A component is specified as a block by assigning it any pre-defined format other than `Normal`.

1. Select the `href` element that has been created as an image (not the one created as an Input Field) by clicking its start tag.
2. From the drop-down Pre-defined Format menu in the toolbar, select `Block (div)`. This makes the image display as a block, and enables the Block Style Window.
3. Place the cursor before the `CompanyLogo` start tag and press Backspace to remove the line break.
4. In the Block Style Window, select the Layout tab (shown below), and specify `border-bottom-style=solid`; `border-bottom-width=medium`; and any color for `border-bottom-color`.

 © 2005 Altova GmbH

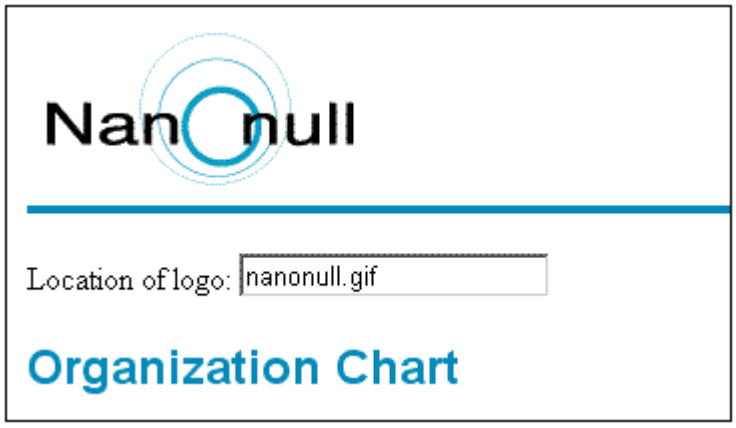


5. Check the Authentic or HTML Preview, which should look like this:



The logo is flush with the left margin of the page and with the bottom border. Some padding would improve the display.

6. In the Design Window, select the `href` element again, and in the Layout tab of the Block Style Window set the `padding-left` and `padding-bottom` properties to 10px. The Authentic and HTML Previews will now look like this:



Now format the components currently in the Design Document as you wish using the pre-defined formats and the Text Style and Block Style Windows.

3.6 Inserting a dynamic SPS table

A dynamic SPS table (hereafter called dynamic table) organizes the child nodes (elements and attributes) of a given element into a table. The table can have headers and footers which can have static or dynamic content. The entire structure of the table, including that of headers and footers, can be modified graphically in the Design Window. One dimension of the table (say the columns) consists of the child elements of the node that is being created as a table; the other (the rows) would then consist of the occurrences of the selected element. Thus the first row would show the child nodes of the first occurrence; the second row the child nodes of the second occurrence; and so on. Dynamic tables are ideal for presenting the content of repeating data structures.

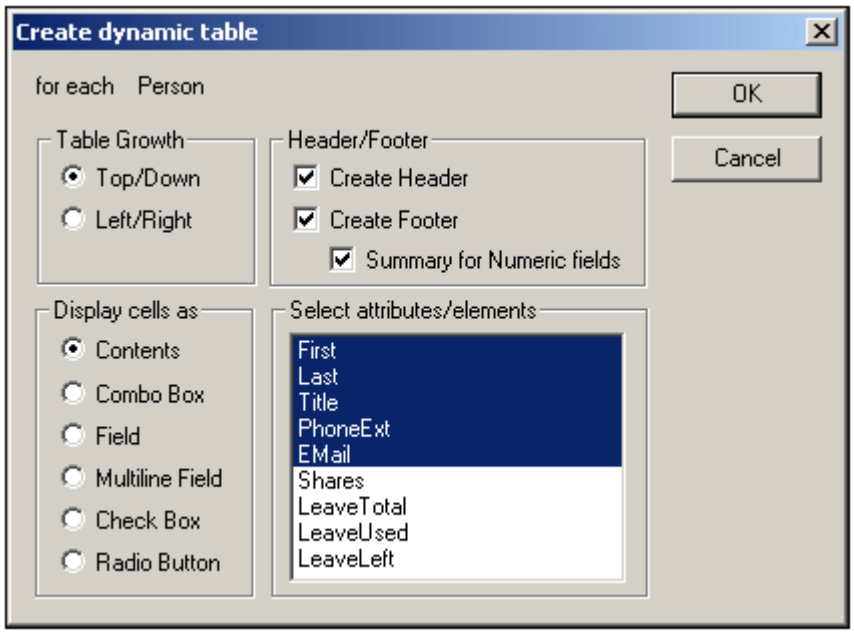
The structure and formatting of a dynamic table are defined by you, the designer of the StyleVision Power Stylesheet. The Authentic View user is not allowed to modify the formatting of the table, but can insert, append, or delete occurrences of the element that has been created as the table, i.e. modify one dimension of the table structure. The Authentic View user can also edit the XML content (dynamic content) of the table (except for Auto-Calculation nodes (you specify the input parameters of the Auto-Calculation) and nodes that you specify as non-editable).

Inserting the `Person` element as a dynamic SPS table

The `Department` element has two child elements: `Name` and `Person`. The `Person` element has several child elements: `First`, `Last`, `Title`, etc. Since there are a number of people in a given department, the `Person` element will occur several times. The `Person` element is a repeating data structure.

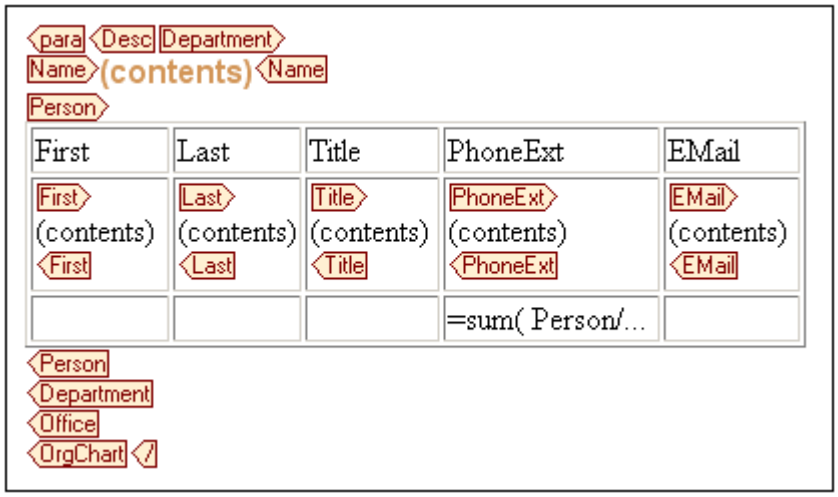
In this part of the tutorial, you will first insert the department's name and, below it, a table containing the names and other details of the persons in that department. Do the following:

1. Drag the `//Department/Name` element from the schema tree, and drop it just after the end tag of the Desc element. Create it as contents, and give it suitable text formatting.
2. Drag the `//Department/Person` element from the schema tree, and drop it just after the end tag of the `//Department/Name` element.
3. In the pop-up menu that appears, select Create as Table. The Create dynamic table dialog appears:



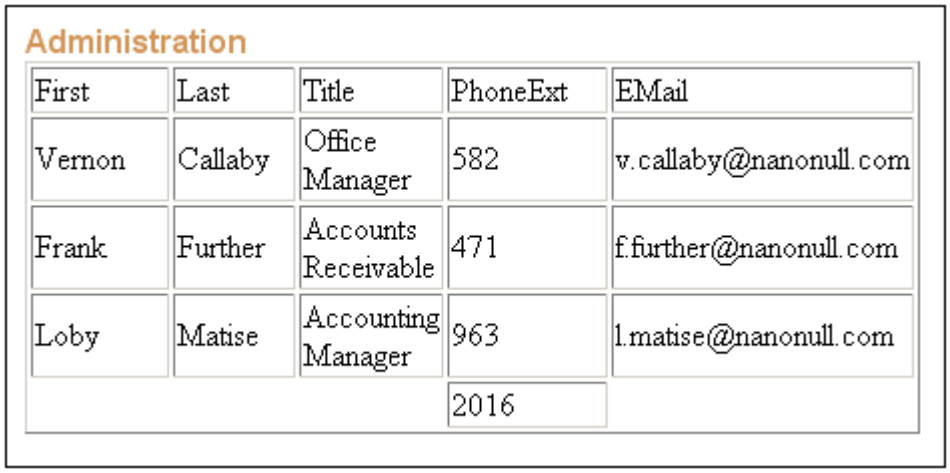
 © 2005 Altova GmbH

4. All the child element fields are highlighted by default, and will be created as columns. Deselect `Shares`, `LeaveTotal`, `LeaveUsed`, and `LeaveLeft`. The default setting creates a header row for the table, in which the header for each column is the name of the respective child element. If the table grows from top to bottom, you can also create a footer with or without sums of numeric columns. Check the Create Footer and Summary for Numeric Fields check boxes. Click OK to create the table.
5. To adjust the vertical spacing, put a line break before the `//Department/Name` element and after the `//Department/Person` element. This part of the Design Document will look like this:



Notice that the column headers are the names of the child elements of the `Person` element, and that a footer has been created with a sum of the numeric column for telephone extensions.

In the HTML Preview, the corresponding output for the first department will look like this:

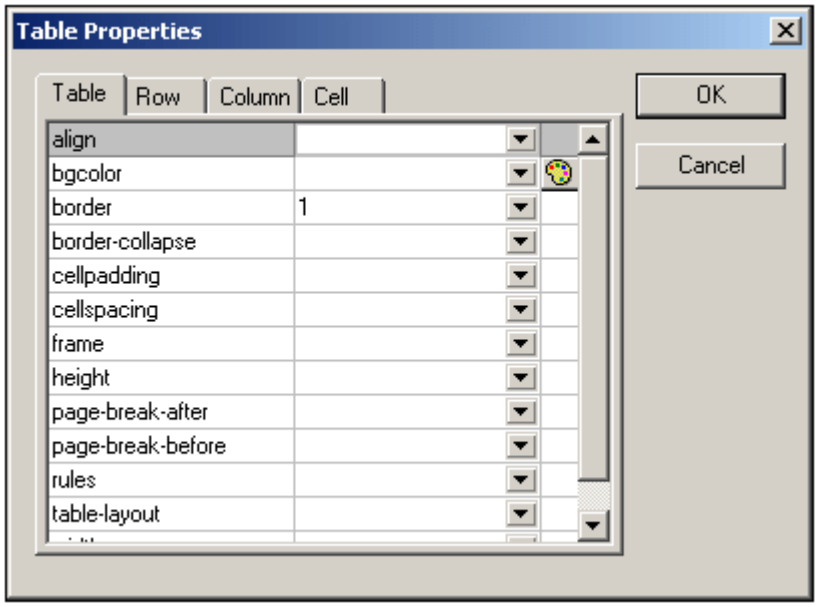


Administration

First	Last	Title	PhoneExt	EMail
Vernon	Callaby	Office Manager	582	v.callaby@nanonull.com
Frank	Further	Accounts Receivable	471	f.further@nanonull.com
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com
			2016	

Modifying table properties

To modify the table properties, place the cursor anywhere in the table, and click  or **Table | Table Properties**. The following dialog pops up:



Make the following modifications:

1. For the entire table, in the Table tab, set border to 1, cellpadding to 3, cellspacing to 0, and width to 100%.
2. To give the header row a background color, place the cursor in the header row, click **Table | Table Properties**, and, in the Row tab, specify a bgcolor.
3. Now give the footer row a background color using the same method.
4. To modify the column widths, place the cursor in the first column, click **Table | Table Properties**, and, in the Column tab, specify a width of 15%. Repeat the steps for each column, using the following widths: second column=15%, third column=25%, fourth column=15%, fifth column=30%.
5. If you wish to assign the column headers some text formatting, mark each header, and assign properties in the Text Style Window. You can also set text alignment properties (in the Text tab of the Block Style Window).

   The Authentic and HTML Previews should look something like this:

Administration

First	Last	Title	Ext	Email
Vernon	Callaby	Office Manager	582	v.callaby@nanonull.com
Frank	Further	Accounts Receivable	471	f.further@nanonull.com
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com
			2016	

Advanced table formatting
Later in the tutorial, we will increase the complexity of this dynamic table, at which stage you will learn advanced table formatting. In the Using Auto-Calculation section, you will append columns

 © 2005 Altova GmbH

to the dynamic table and split the header into two rows. In Creating the Table Footer, you will add content to the footer and format it.

Note:

1. Footers must be created when the table is first created. **You cannot append a row and treat it as a footer.** This is because in the Design Document you will be appending rows to the construct (usually a single row) that defines how each occurrence is displayed. This results in the appended row/s being placed after each occurrence of the element that was created as the table, creating an alternation of occurrence row and appended row/s (not a footer of appended row/s). We will format the footer in the Creating the Table Footer section later in the tutorial.
2. When you set column widths as percentages, they must add up to 100%. To avoid possible errors, set the table width to 100%.
3. If there are column spans (colspans) in your table, you might need to assign width to individual cells in columns rather than to the columns themselves. You can do this by placing the cursor in a cell, and assigning width properties in the Cell tab of the Table Properties dialog.
4. You can also modify column widths by dragging column borders. However, it is advisable not to mix different measurement formats (percentages and absolute values) or different measurement units (px and mm).
5. You can assign properties for cells in the Block Style Window also (in addition to the Cell tab of the Table Properties dialog). See also Formatting static and dynamic tables.

3.7 Inserting a static SPS table

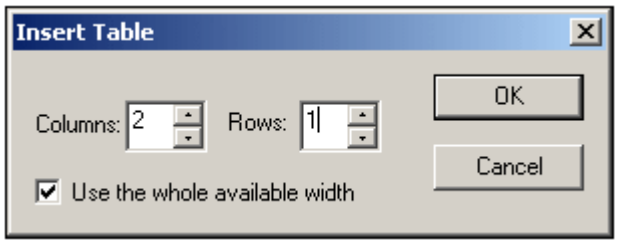
A static SPS table is a StyleVision Power Stylesheet component that you use for improved visual presentation. The cells of a static SPS table may contain either dynamic content (from the XML document) or static content (text, images, etc.).

The structure and formatting of a static SPS table are defined by you, the designer of the StyleVision Power Stylesheet. The Authentic View user is not allowed to modify either the structure or formatting. The Authentic View user is allowed to edit only the dynamic (XML) content of the table (excepting for Auto-Calculation nodes and nodes that you specify as non-editable).

Inserting a static table with miscellaneous content
The objective is to create a table that presents the contact details of each office in a visually attractive way. Let us put the address in the first column of a two-column table, and the telephone and fax numbers and the e-mail address in the second column. We start with a table of two columns and one row, and edit it as we go along.

To create the static SPS table, do the following:

1. In the Design Window, place the cursor after the end tag of the `//Office/Name` element (by clicking the element's end tag and pressing Cursor Right). You get a blinking cursor after the end tag of `//Office/Name`.
2. Click  , or select **Table | Insert Table**. This causes a static SPS table to be inserted at the cursor location. The Insert Table dialog (shown below) appears.

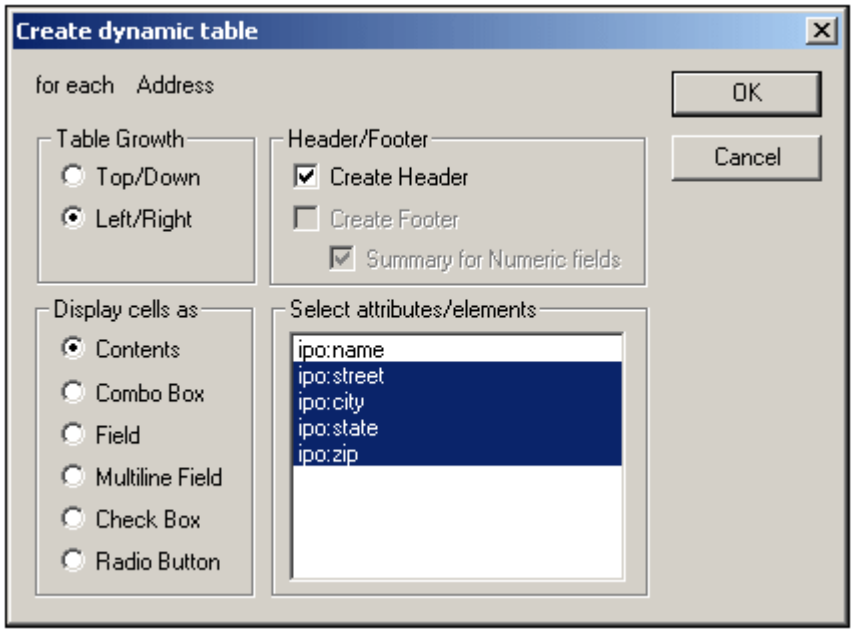


3. In the Insert Table dialog, specify a table of 2 columns and 1 row, and click OK. A 2x1 table is inserted in the Design Document (shown below).

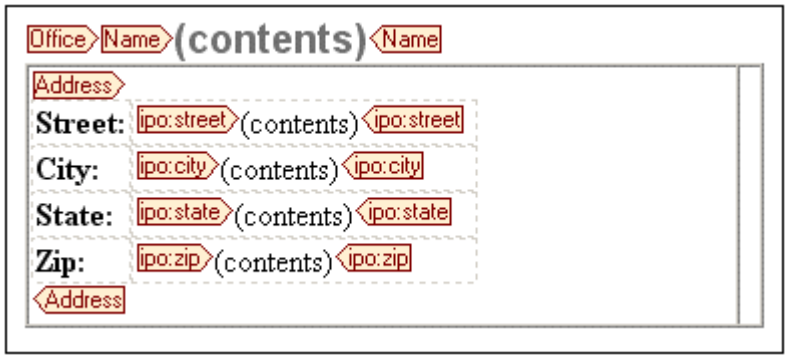


4. Drag the `Address` element from the schema tree and drop it into the first cell of the table.
5. Select Create Table from the popup. This opens the "Create dynamic table" dialog box.

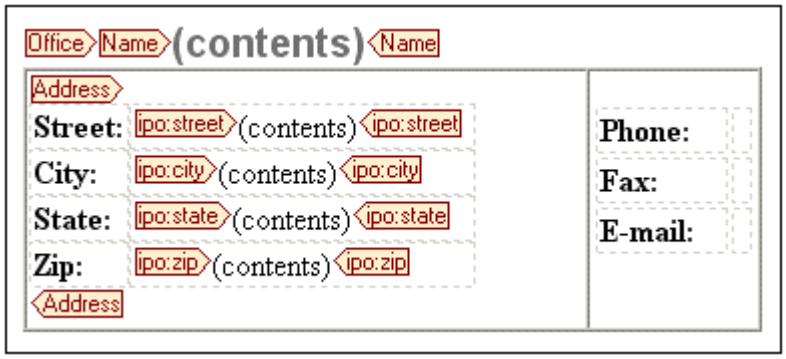
 © 2005 Altova GmbH



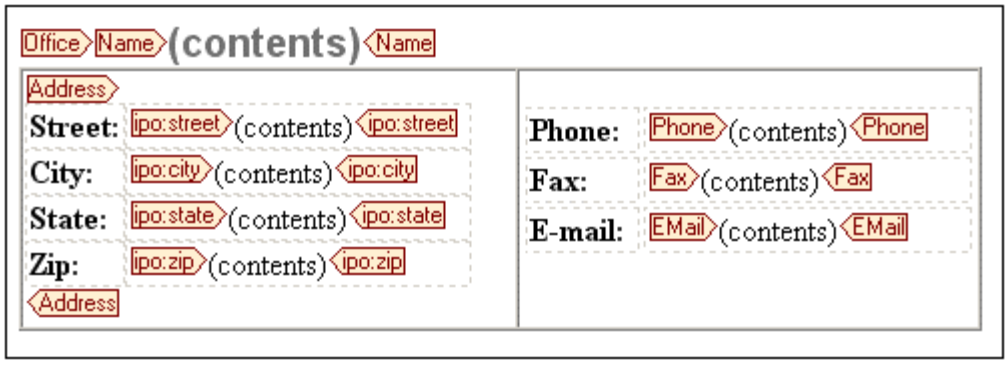
6. Click the "ipo:name" entry (in the Select attributes/elements box) to deselect it.
7. In the Table Growth option, select the Left/Right radio button. Notice that the option for creating a Footer is automatically disabled; it is enabled only when the dynamic table grows top-down. Click OK. This inserts a dynamic SPS table in the first cell of the static SPS table you just created.
8. Set the cellspacing for the outer static table to 0 and the border for the inner dynamic table to 0 (in the Table Properties for each). Then format the field names as shown below. Your Design Document will now look like this:



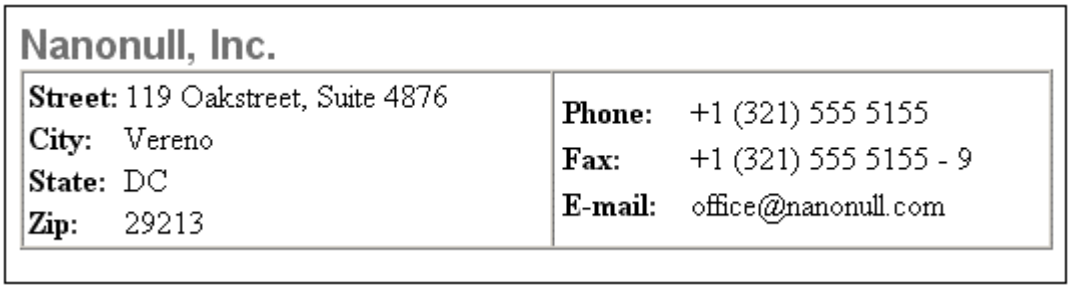
9. Place the cursor in the second column of the outer static table, and click  or select **Table | Insert Table**. In the Insert Table dialog, define the table as having 2 columns and 3 rows. This creates a nested static table in the second column of the outer static table.
10. Click in the first column of the new static table, and enter the text Phone, Fax, and E-mail, respectively in each row of the first column as shown below, and make the font bold. In the Table Properties dialog, set the table border to 0.



11. Drag the `Phone` element from the schema tree, and drop it into the cell adjacent to that with "Phone" in it. Select Create Contents from the popup.
12. Do the same for the `Fax` and `Email` elements: drop these elements into the appropriate cells, selecting "Create Contents" in both cases.



Check the Authentic or HTML Preview. The static SPS table should look like this:



Adjusting the vertical alignment
In the HTML Preview, notice that the text in the two columns are not vertically aligned. You can ensure alignment by vertically aligning both columns of the outer static table, and specifying the same cellspacing value for both nested tables.

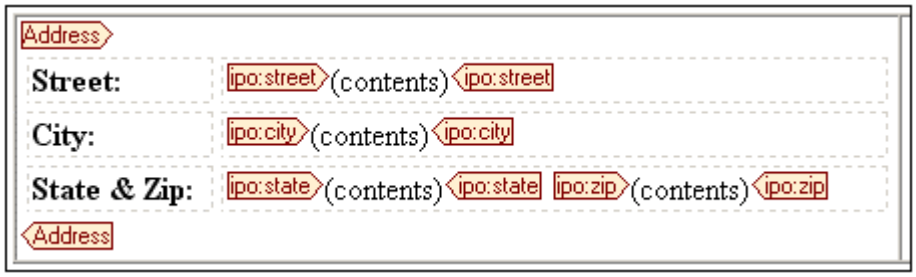
1. Place the cursor anywhere in the first column of the static table, taking care not to place it in the nested table in that column.
2. Click  or **Table | Table Properties**, and select the Column tab.
3. For the `valign` property, select the value `top`.
4. Repeat the above three steps for the second column of the outer static table.
5. Specify cellspacing of 4px for both nested tables (in the Table tab of the Table Properties dialog for each table).
6. Check the Authentic or HTML Preview to see the effect of your changes. The text in both columns should now be aligned.

 © 2005 Altova GmbH

Joining cells
The objective is to have the State and Zip fields appear in a single row (so that both columns have three-row tables). Do the following:

1. Place the cursor in the cell containing the text "Zip:".
2. Click  or **Table | Join cell above**.
3. Change the text to "State & Zip:".
4. Place the cursor in the cell containing the contents of the `ipo:state` element.
5. Click  or **Table | Join cell below**.
6. Put a space between the two `(contents)` placeholders in the cell.

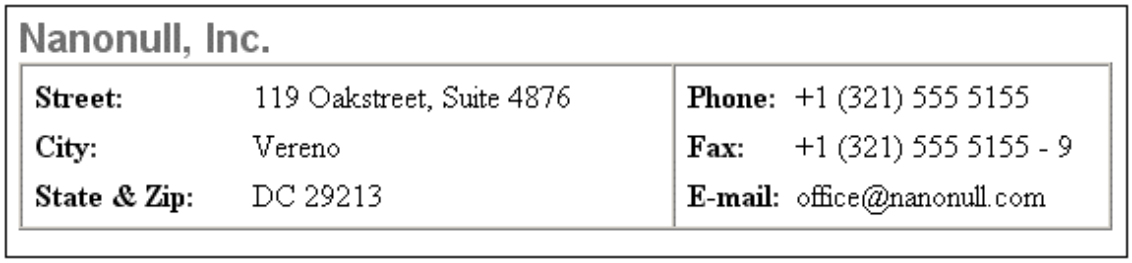
The nested dynamic table in the Design Document will now look like this:



Specifying the widths of columns
Check the Authentic and HTML Previews so you can judge the optimal column widths to use. After checking the previews, do the following:

1. Place the cursor anywhere in the first column of the outer static table, but not in the nested dynamic table, and click  or **Table | Table Properties**.
2. In the Table tab, set (table) `width` to 100%.
3. In the Column tab, set (column) `width` to 60%.
4. Place the cursor in the second column of the outer static table, but not in the nested static table, and click  or **Table | Table Properties**.
5. In the Column tab, set (column) `width` to 40%.
6. For the nested dynamic table, set the table width to 100%, width of first column to 30%, and width of second column to 70%.
7. For the nested static table, set table width to 100%, width of first column to 25%, and width of second column to 75%.

The Authentic and HTML Previews of the static SPS table should look like this:


Nanonull, Inc.
Street: 119 Oakstreet, Suite 4876
City: Vereno
State & Zip: DC 29213
Phone: +1 (321) 555 5155
Fax: +1 (321) 555 5155 - 9
E-mail: office@nanonull.com

3.8 Inserting bookmarks and hyperlinks

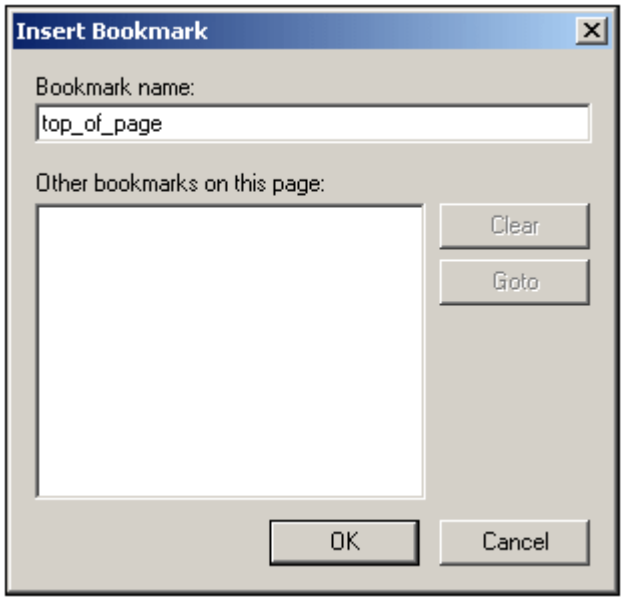
You can create bookmarks (anchors) and hyperlinks in the HTML output. In the StyleVision Power Stylesheet, you can insert a bookmark at any location in the Design Document. You can then insert a hyperlink to that bookmark in the document or to any URL. In the output (HTML), the anchor (for the bookmark) and the hyperlink (to internal anchors and external locations) is created. Note, however, that linking is not available in Authentic View (since this is a view of the source XML document).

In this section you will:

- Create a bookmark at the top of your document.
- Create a hyperlink at the bottom of the document which links back to the bookmark.
- Create a hyperlink to a Website.
- Create a hyperlink to a pre-addressed e-mail.

Creating a bookmark at the top of your document

1. Place the cursor just after the start tag of the `OrgChart` element, and right-click.
2. From the popup menu that appears, select **Insert | Bookmark**. The following dialog appears:



3. Give your bookmark a name, such as `top_of_page`, and click OK. The bookmark is created and is indicated by a flag.

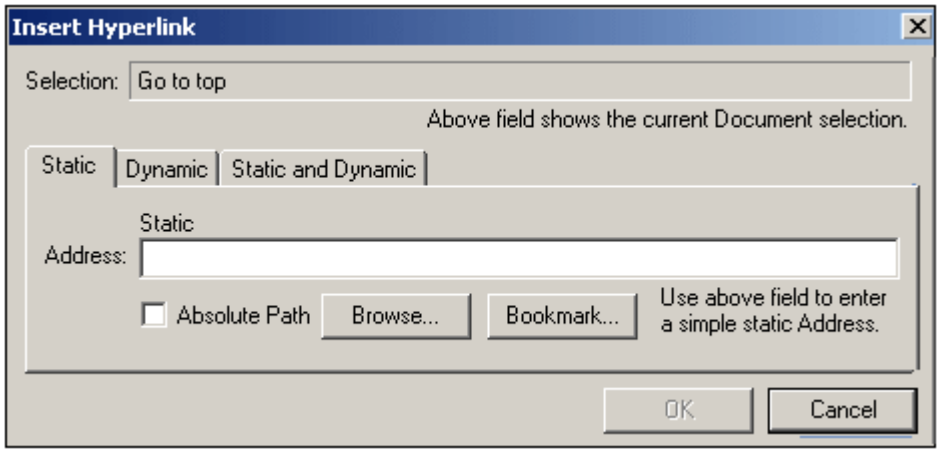


Inserting a hyperlink to a bookmark

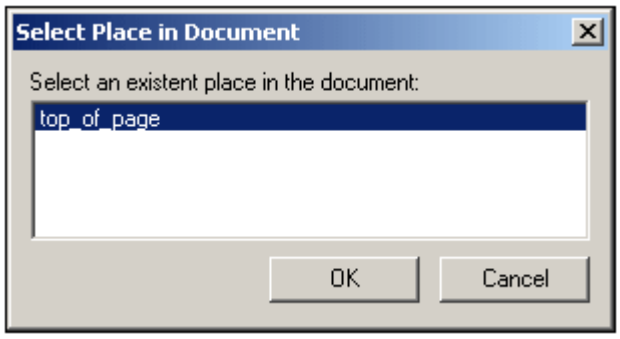
1. Place the cursor just before the end tag of the `OrgChart` element, and type in "Go to top".
2. Format this text in the Text Style Window, and modify the line breaks at the bottom of the document to your satisfaction.
3. Highlight the text and right-click.

 © 2005 Altova GmbH

4. From the popup menu that appears, select **Insert | Hyperlink**. The Insert Hyperlink dialog appears:



5. Since you want to link to a bookmark, click the Bookmark button. The following dialog appears with all the bookmarks in the document listed.



6. Select the bookmark you want, and click OK. The bookmark appears as the static address of the hyperlink in the Insert Hyperlink dialog, i.e. the hyperlink is explicitly specified in the StyleVision Power Stylesheet and is not contained in the XML document.
7. Click OK. The hyperlink is created.
8. In the previews, go to the end of the document, and click the hyperlink "Go to top". The view will change to the top of the document.

Inserting a hyperlink to a Website

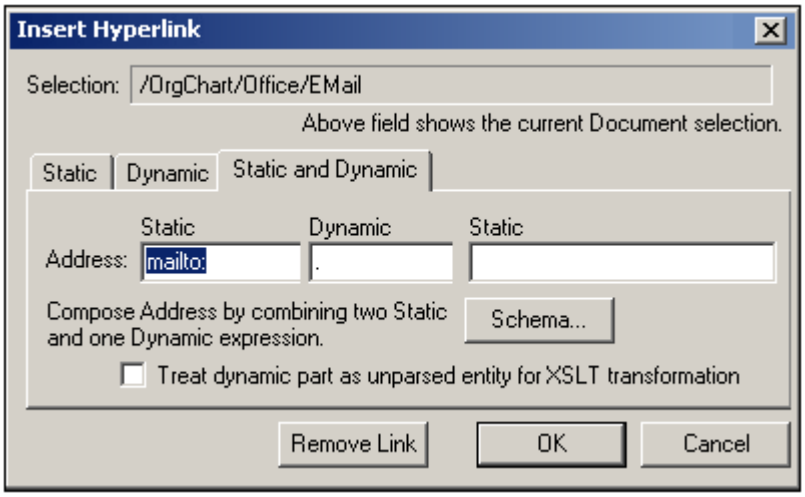
1. Right-click the logo image, and in the popup menu that appears select **Insert | Hyperlink**.
2. In the Insert Hyperlink dialog that appears, enter `http://www.nanonull.com/` as the static address of the hyperlink target.
3. Click OK. The hyperlink is created.
4. In the HTML document, click the logo image. The `www.nanonull.com` Web page appears in the view.

Inserting a hyperlink to a pre-addressed e-mail

The objective is to create a link to an e-mail that has in its address field the correct office or employee address.

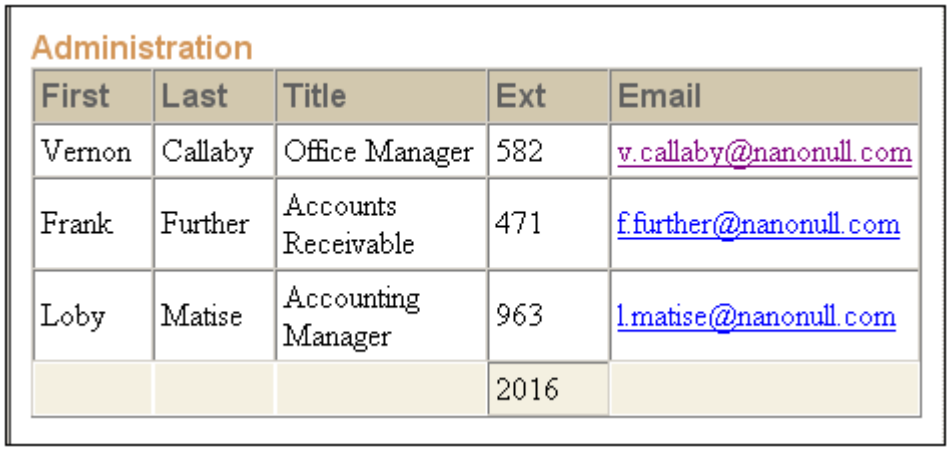
1. In the Design Document, right-click the `/OrgChart/Office/EMail` element (or its `(contents)` placeholder), and in the popup menu that appears select **Insert | Hyperlink**.

2. In the Insert Hyperlink dialog, click the **Static and Dynamic** tab.
3. The address consists of a static part and a dynamic part. Fill these in as shown in the figure below. (Don't miss the period in the dynamic part of the address; this specifies that the content of the current node (shown in the Selection field) is to be processed.)



4. Click OK. The hyperlink is created. The static `mailto:` part of the address is the protocol part of the URL. The dynamic part of the address is the content of the `//Office/EMail` element, which is the e-mail address of each office, respectively.
5. Right-click the `//Person/EMail` element (or its `(contents)` placeholder), and follow the same steps as for the previous (office) e-mail address. Each person's e-mail address is now a hyperlink, which, on being clicked, generates an e-mail with that person's e-mail address in the Address field of the e-mail.
6. In the HTML document, click a few e-mail addresses for various offices and persons. For each, a new e-mail with the correct address will be opened.

The HTML Preview is shown below:



Administration

First	Last	Title	Ext	Email
Vernon	Callaby	Office Manager	582	v.callaby@nanonull.com
Frank	Further	Accounts Receivable	471	f.further@nanonull.com
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com
			2016	

 © 2005 Altova GmbH

3.9 Using Auto-Calculation

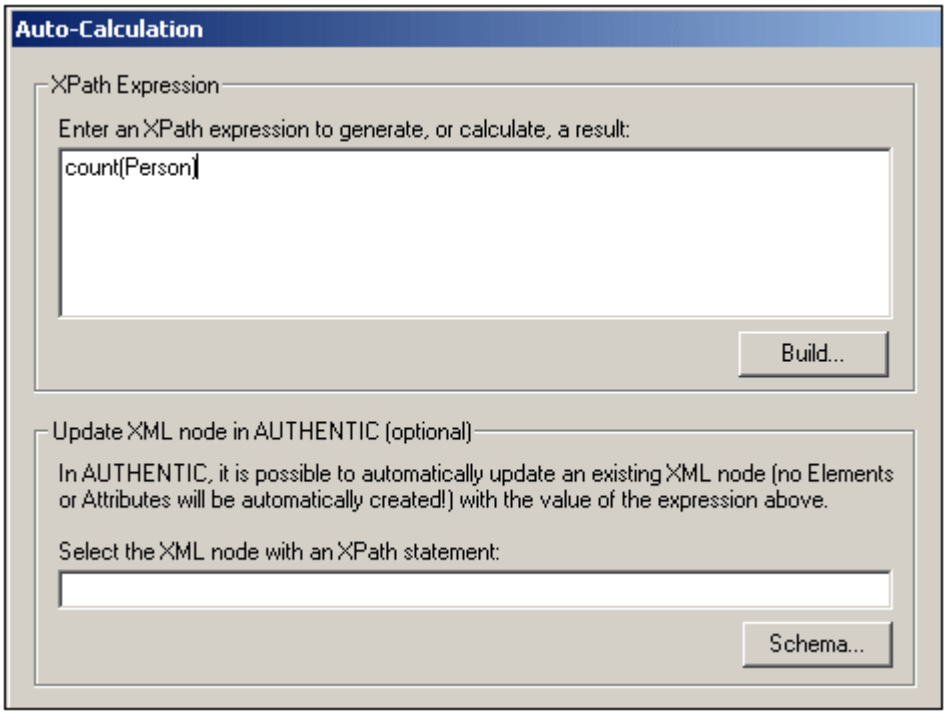
StyleVision includes an Auto-Calculation function, which enables you to display the result of a calculation in Authentic Viewand HTML output. The result is calculated using an XPath expression. It changes dynamically when the value of a node used in the XPath expression changes. The Auto-Calculation function is a useful mechanism for including dynamic XML content and calculations based on dynamic content in the output.

In this tutorial, you will use the Auto-Calculation feature to add the following information to your StyleVision Power Stylesheet:

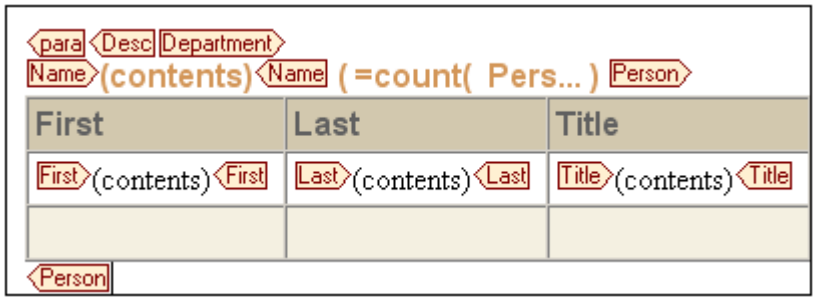
- The number of employees in each department
- The number of departments and employees in each office
- The leave available to each employee, which is the total leave allowed each employee minus the leave used up.

Auto-calculating employees in each department

1. Place the cursor after the end tag of the `//Department/Name` element.
2. Right-click, and from the popup menu select **Insert | Auto-Calculation | Value**. The Auto-Calculation dialog appears:

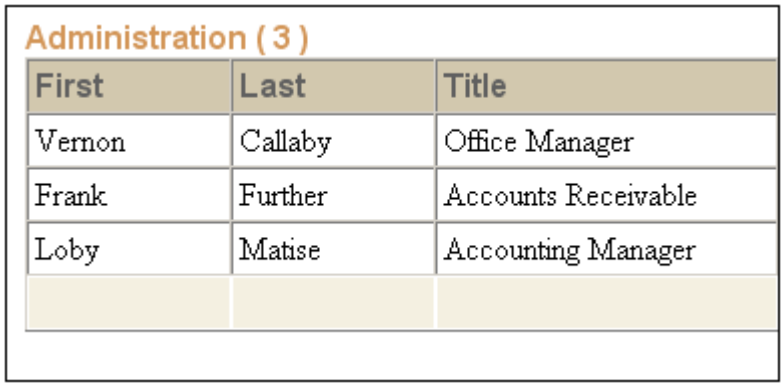

Auto-Calculation
XPath Expression
Enter an XPath expression to generate, or calculate, a result:
count(Person)
Build...
Update XML node in AUTHENTIC (optional)
In AUTHENTIC, it is possible to automatically update an existing XML node (no Elements or Attributes will be automatically created!) with the value of the expression above.
Select the XML node with an XPath statement:
Schema...

3. Build the XPath expression by clicking the **Build...** button. Notice that the context node is `Department`, indicated by its being highlighted in gray. Now build the XPath expression: `count(Person)` (by double-clicking the required node and XPath operators and functions). Alternatively, you can type the XPath expression directly into the XPath expression field.
4. Click OK. The Auto-Calculation placeholder is inserted.
5. Format the placeholder as desired using the Text Style Window. This part of the Design Document will look something like this:



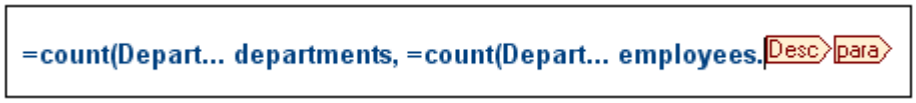
Note that the parentheses in the figure above have been inserted as a design item; they are not placeholder delimiters.

6. Check the HTML Preview. The HTML Preview of the Auto-Calculation result for the first department is shown below:



Auto-calculating departments and employees in each office

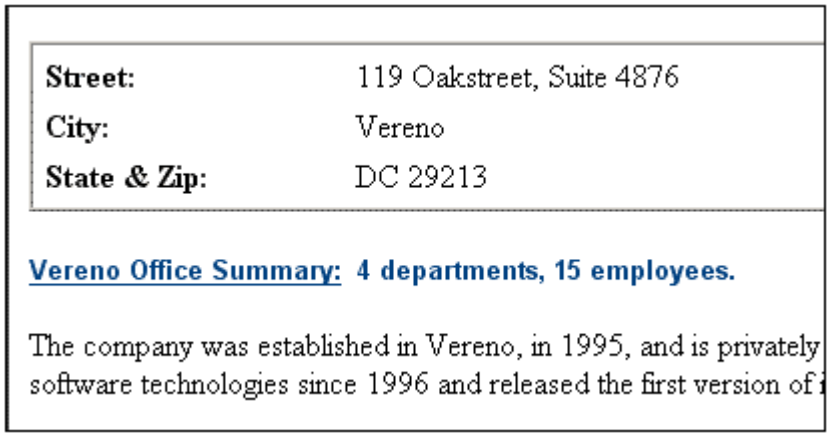
1. Place the cursor in front of the start tag of the `//Office/Desc` element. (Note that the context node will therefore be `Office`.)
2. Use the method above to create Auto-Calculation values for the number of departments (`count(Department)`) and the number of employees in the department (`count(Department/Person)`).
3. Add the word "departments" after the auto-calculation for the number of departments, and "employees" after the auto-calculation for the number of employees.
4. Format the line as required.



5. Insert the `//Address/ipo:city` element as contents before the Auto-Calculation placeholders, and append the words "Office Summary" so that the line in the StyleVision Power Stylesheet now looks similar to this.



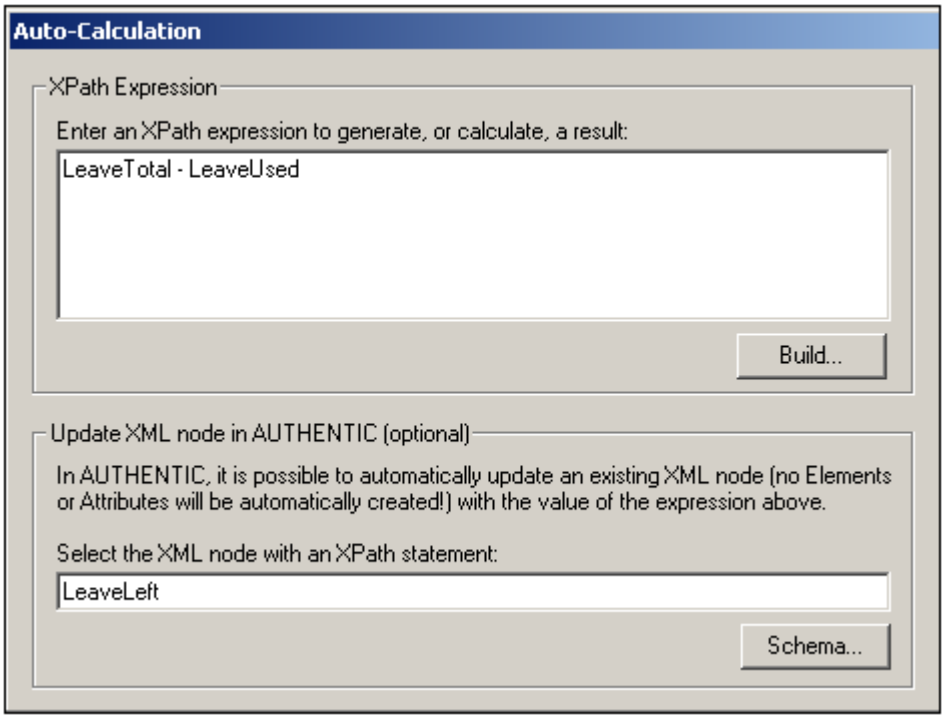
6. Format the inserted components as required. The corresponding line in the Authentic and HTML Previews should look something like this:


Street: 119 Oakstreet, Suite 4876
City: Vereno
State & Zip: DC 29213
Vereno Office Summary: 4 departments, 15 employees.
The company was established in Vereno, in 1995, and is privately
software technologies since 1996 and released the first version of

Including the leave details
The objective is to allow the Authentic View user to edit the `LeaveTotal` and `LeaveUsed` entry for each employee. The `LeaveLeft` is auto-calculated (`LeaveTotal` minus `LeaveUsed`), and displayed.

Appending the LeaveTotal, LeaveUsed, and LeaveLeft columns

1. Place the cursor anywhere in the dynamic table for the `Person` element and click **Table | Append Column** or the  icon thrice to create three new columns.
2. Type in "Leave Total", "Leave Used" and "Leave Left" as the headers of the three appended columns, respectively.
3. Format the headers appropriately.
4. Drag the `LeaveTotal` and `LeaveUsed` elements from the schema tree into the Leave Total and Leave Used columns respectively, and create them as contents.
5. In the Leave Left column, create an Auto-Calculation (`LeaveTotal-LeaveUsed`). See figure below.


Auto-Calculation
XPath Expression
Enter an XPath expression to generate, or calculate, a result:
LeaveTotal - LeaveUsed
Build...
Update XML node in AUTHENTIC (optional)
In AUTHENTIC, it is possible to automatically update an existing XML node (no Elements or Attributes will be automatically created!) with the value of the expression above.
Select the XML node with an XPath statement:
LeaveLeft
Schema...

6. Set `LeaveLeft` as the node to be updated. This updates the XML value of `LeaveLeft` whenever the Authentic View user edits the `LeaveTotal` or `LeaveUsed` value.
7. To accommodate the new column, change the font size of all the element contents in the cell to 10pt.
8. Change the table properties as follows: Place the cursor by turn in each of the three new columns, and for each column, in the Table Properties dialog (Column tab), set `align` to center. Reset all the column widths as follows: First = 10%; Last = 13%; Title = 20%; Ext = 7%; EMail = 26%; Leave Total = 8%; Leave Used = 8%, Leave Left = 8%.

The Authentic and HTML Previews will look like this:

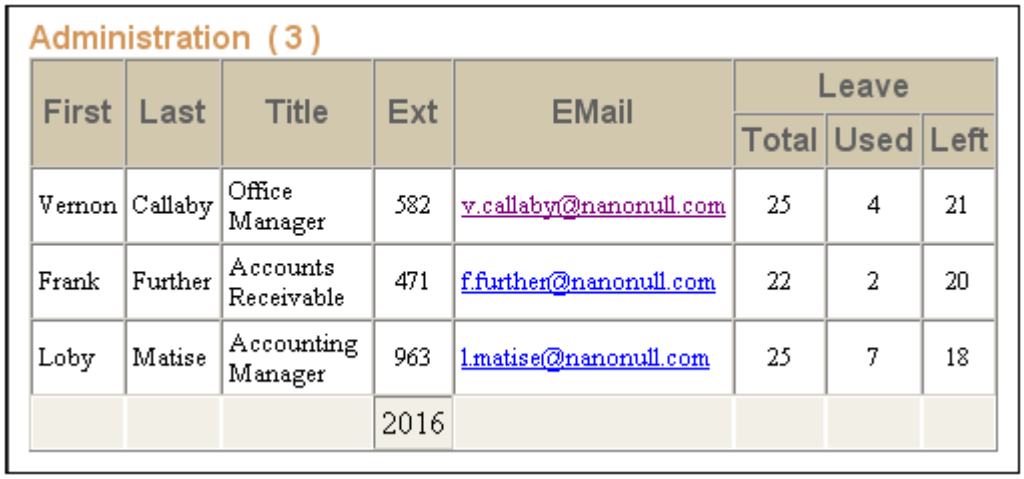
Administration (3)

First	Last	Title	Ext	EMail	Leave Total	Leave Used	Leave Left
Vernon	Callaby	Office Manager	582	v.callaby@nanonull.com	25	4	21
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	22	2	20
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	25	7	18
			2016				

Notice that although the Leave fields are numeric fields, no summary has been created for these columns in the footer. This is because the Leave columns were appended to the table after the table was created. (If you wish to add a summary for a numeric field after the footer has been created, insert an auto-calculation for that column in the footer.)

Restructuring the table header
The header can be restructured by splitting and joining cells. Content of header cells can also be modified, and can include both static and dynamic content. Restructure as follows:

1. Place the cursor in the header cell "Leave Total", and click  or **Table | Split Cell Vertically**.
2. Do the same for the Leave Used and Leave Left headers.
3. Place the cursor in the cell containing the text "Leave Left", and click  or **Table | Join Cell Left**.
4. Click  or **Table | Join Cell Left** once more.
5. Now edit the text in these header cells as shown in the HTML Preview below.



Administration (3)

First	Last	Title	Ext	EMail	Leave		
					Total	Used	Left
Vernon	Callaby	Office Manager	582	v.callaby@nanonull.com	25	4	21
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	22	2	20
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	25	7	18
			2016				

General Note:

- Auto-Calculation can also be created as an Input Field, but the node is non-editable in Authentic View—as is the case with all Auto-Calculations. The Input Field in this case is intended only as a presentational device.
- When an existing XML file is opened in Authentic View, Auto-Calculations are carried out, and the correct Auto-Calculation value is displayed in Authentic View. However, no value in the XML document will be modified. It is only when the user edits a node that is used in the XPath expression of an Auto-Calculation that XML values affected by such an Auto-Calculation are changed.
- When the Authentic View user appends a new `Person` element, all mandatory children are created with empty content. This is because adding mandatory children is the default setting for the `Person` node; it can be changed in the Authentic Node Settings dialog for the `Person` node (select the `Person` element in the Design Window and click **Authentic | Node settings...**). Any non-mandatory child of the `Person` element is indicated with the prompt `Add` *`element-name`*.

3.10 Using Conditional Templates

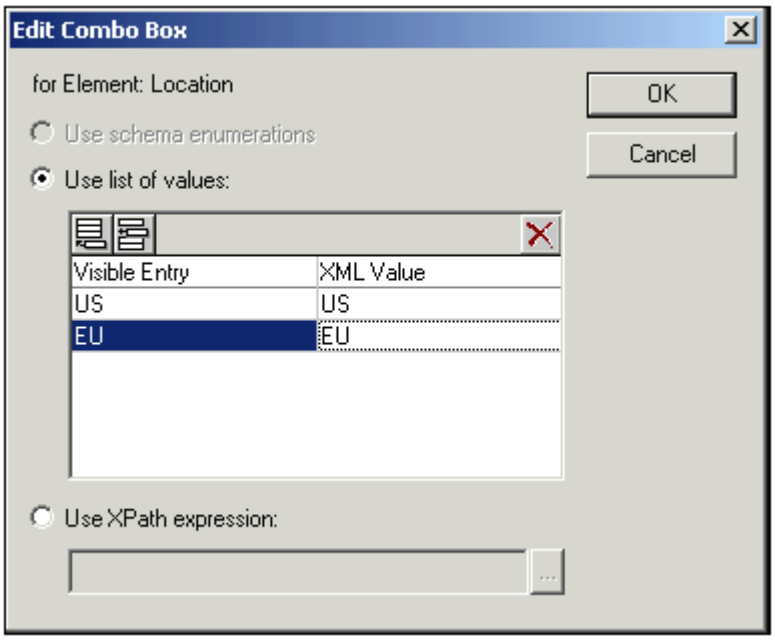
A conditional template defines how a component is to be processed in the event that a specified condition is fulfilled. Multiple conditions can be specified for a component, thus allowing you to provide alternative ways to process a component for a range of conditions. In this section of the tutorial, you will create conditional templates for the following situations:

- Depending on whether an office is located in the US or Europe, the address field will be structured differently.
- In the Office Summary statement, a condition will be used to select the correct city element depending on whether the US or EU address model is used.
- If an employee is a shareholder, the employee's name will be displayed in bold.

Setting up alternative address content models based on a condition
The objective is to display the appropriate address content model depending on whether the office is located in the US or European Union (EU). You will first create a combo box to accept input for the `Location` element (US or EU), and then create two conditional templates. One conditional template defines that, if the `Location` element has a value of US, the address content model for US locations is used. The other conditional template causes the address content model for EU locations to be used if the value of `Location` is EU.

Include the `Location` element in the StyleVision Power Stylesheet as follows:

1. Put a line above the `//Office/Name` element by making it a block (**Insert | Format | Block (div)**) and by assigning, in the Layout tab of the Block Style Window, suitable values for the following properties: `border-top-color`, `border-top-style`, and `border-top-width`.
2. Insert a line-break after the `//Office/Name` element, and type in "Location: ".
3. Drag the `Location` element from the schema tree, drop it immediately after the text "Location: ", and create it as a combo box. The following dialog appears (without any entries):



4. Click the Append Row icon twice to append two rows in the dialog box. Enter the two values for US and EU as shown in the figure above. The Visible Entry is the entry that will be visible in the combo box and available for selection. The XML Value is the corresponding value that will be entered in the XML file when a Visible Entry is selected.

 © 2005 Altova GmbH

5. Click OK when done. The Design Document will now look something like this:

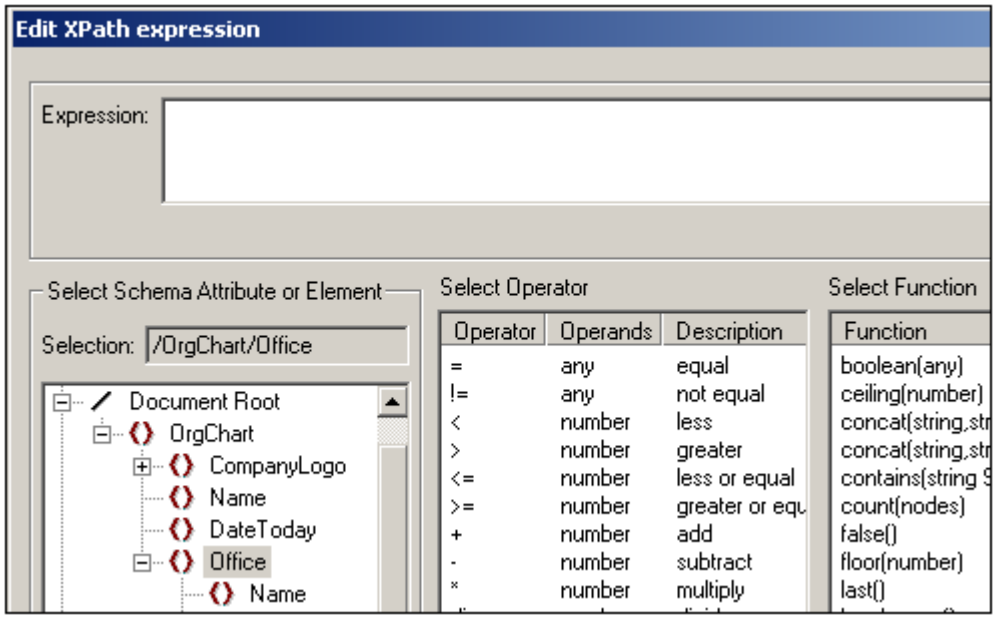


The corresponding output in Authentic Editor View will look like this:



Now specify the conditions that determine the selection of the appropriate address content model. Do the following:

1. Right-click the `Address` element in the static table, and select **Insert | Condition...** to insert a condition for the entire `Address` component in the StyleVision Power Stylesheet. The Edit XPath expression dialog appears:

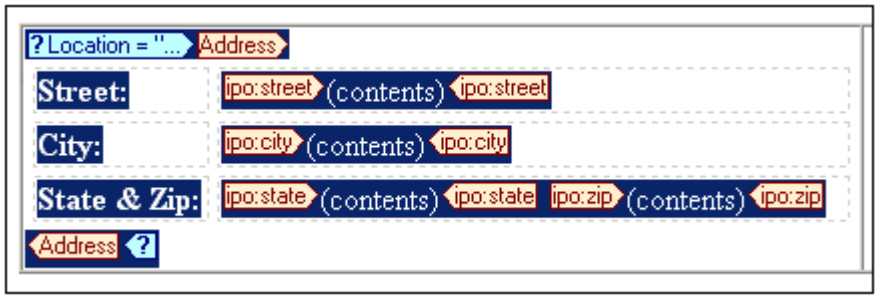


Notice that, in the Select Schema Attribute or Element pane, the `Office` element is highlighted in gray. This indicates that `Office` is the context node, and that the XPath expression will be created relative to it. You can build the XPath expression by double-clicking the required elements/attributes, operators, and functions. Alternatively, you can type the correct XPath expression directly into the Expression field.

2. Build the expression by double-clicking the `Location` element in the Select Schema

Attribute or Element pane. This causes `Location` to be entered into the Expression field. Then click the = operator, and enter `"US"`. The XPath should read: `Location = "US"`.

3. Click OK. The start and end of the conditional template are indicated by cyan tags with a question mark: ?Location = "..." ?. The conditional template in the Design Document will look like this:



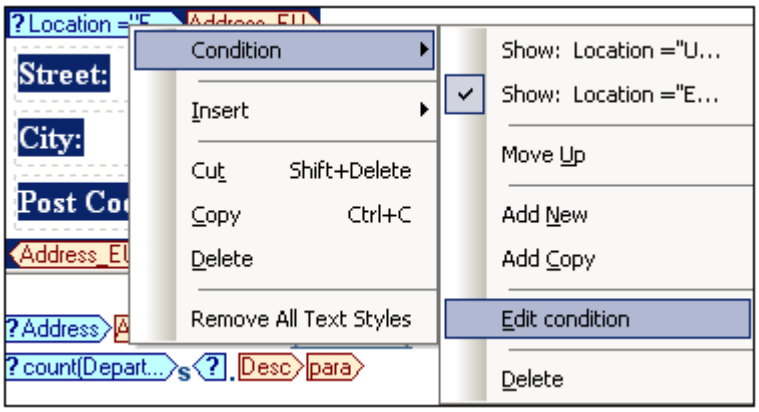
Notice that the entire dynamic table contained by the `Address` element is contained by the tags of the conditional template. This is because you have inserted the condition around the `Address` element.

4. Since the content model of the `Address` element already in the table is that for the US, do not alter it.

This conditional template defines that its contents will be processed and displayed as specified when the `Location` element has content with a value of US.

Now create a conditional template for displaying the content model for European addresses whenever the value of the `Location` element is EU. Do the following:

1. Right-click either of the condition tags of the conditional template you have just created, and select **Condition | Add Copy**.
2. In the Edit XPath expression dialog that appears, enter or build the XPath: `Location = "EU"`.
3. Click OK. You have now created a second conditional template. Only one conditional template is shown at a time. To see which conditional template is currently displayed, right-click, and select **Condition**. The following context menu appears:

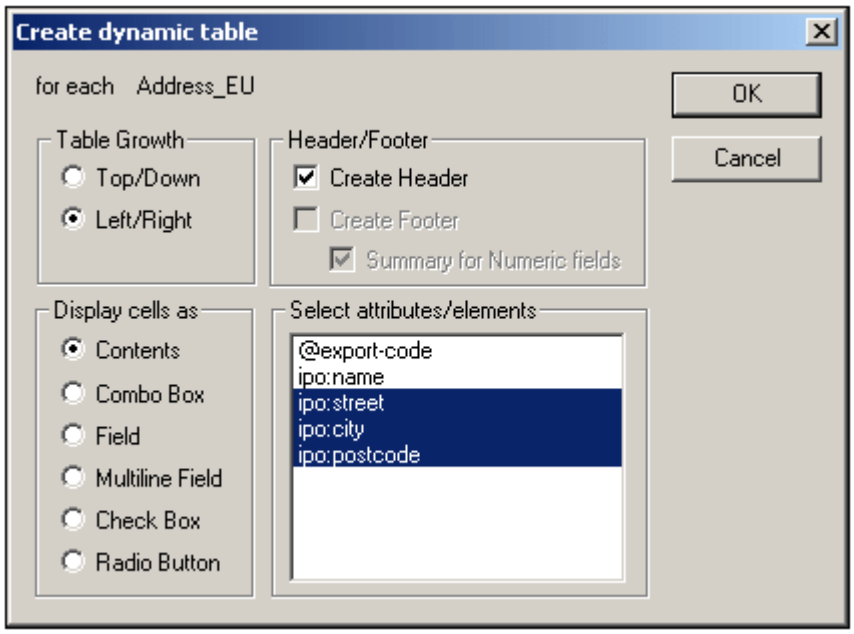


Which conditional template is currently displayed is indicated with a check mark next to it. (If you wish to edit a condition, select **Edit condition**.)

The currently displayed conditional template specifies processing to be carried out when `Location` has a value of EU. The address content model within it, however, is that for the US (since this is a copy of the first conditional template). Change this as follows:

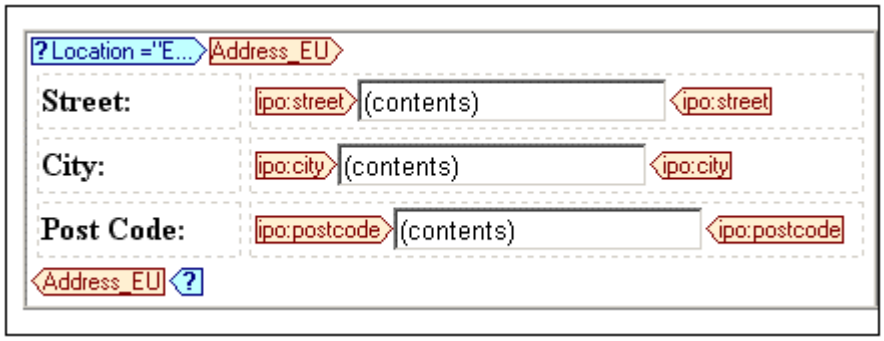
 © 2005 Altova GmbH

1. Delete the entire `Address` element (by clicking either its start or end tag, and pressing Delete).
2. From the schema tree drag the element `Address_EU` into the static table, drop it between the conditional template tags, and create it as a table. The following dialog appears:



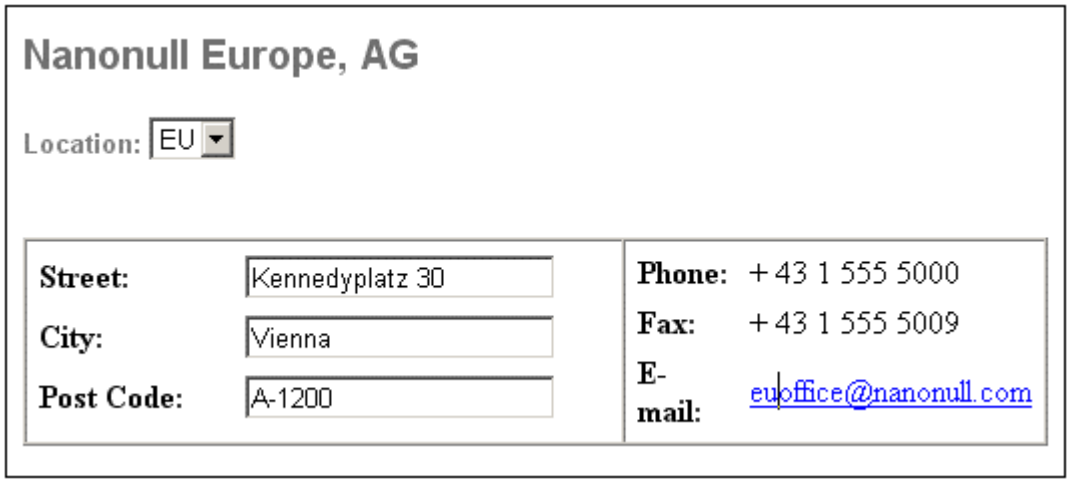
3. Deselect `@export-code` and `ipo:name`. Select Create Header; Table Growth Left/Right; and Display cells as Field.
4. Click OK. The table is created.
5. Format the headers in bold (by highlighting the text and clicking the Bold icon or setting `font-weight` to bold);
6. Set the following table properties: `border` to 0; `cellspacing` to 4px; `table-width` to 100%.

The Design Document looks like this:



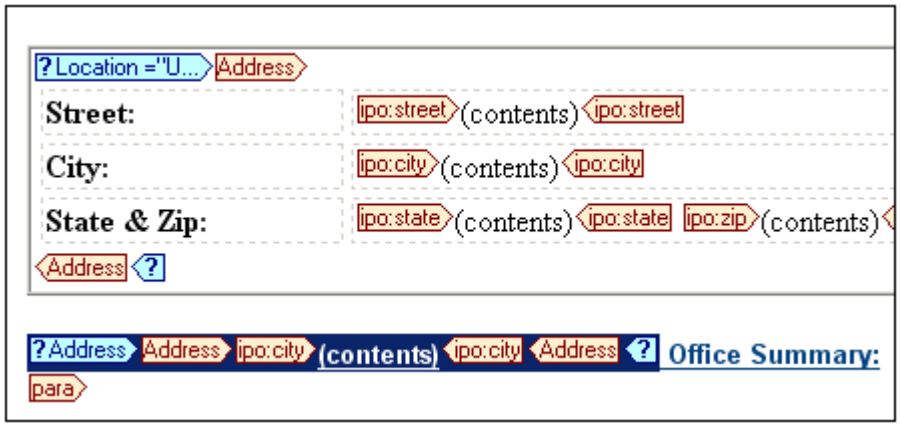
Note that the EU address components have Input Fields; the US address components are created as normal text. However, both types of display can be edited in Authentic View.

The Authentic Editor View looks like this:



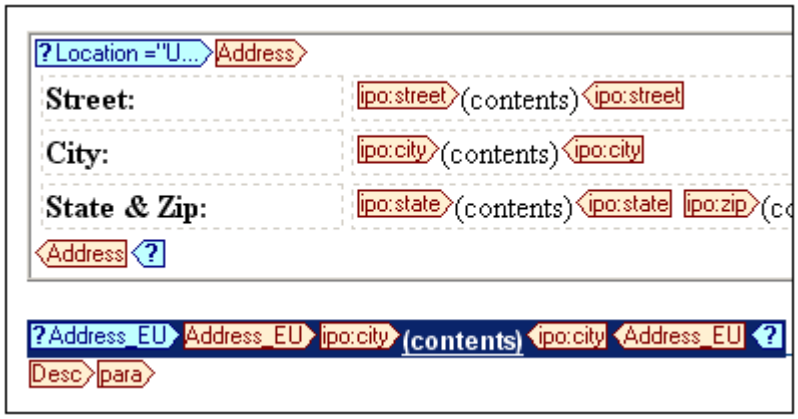
Creating a condition to select the correct city element
In the Office Summary line for each office the city is selected using an Auto-Calculation. Since the city element is a different element in each of the address content models (US and EU address content models), a condition must be created to select the correct city element. Do this as follows:

1. Right-click the `Address` element in the Office Summary line, and select **Insert | Condition...**.
2. In the Edit XPath expression dialog, the context node is `Office`. Enter the XPath: `Address`, and click OK. This creates a conditional template in which the condition tests for the presence of an `Address` child element. The Design Window will look like this:



3. The `//Address/ipo:city` element is the correct child of the `Address` element, so do not alter the content of this conditional template. The result is that when the address element is `Address`, the content of the `//Address/ipo:city` element will be selected.
4. Right-click the conditional template tag, and select **Insert | Condition... | Add New**.
5. In the Edit XPath expression dialog, enter the XPath: `Address_EU`, and click OK.
6. Drag the `//Address_EU/ipo:city` element into the Design Window, and create it as contents of the new conditional template. The Design Window will look like this:

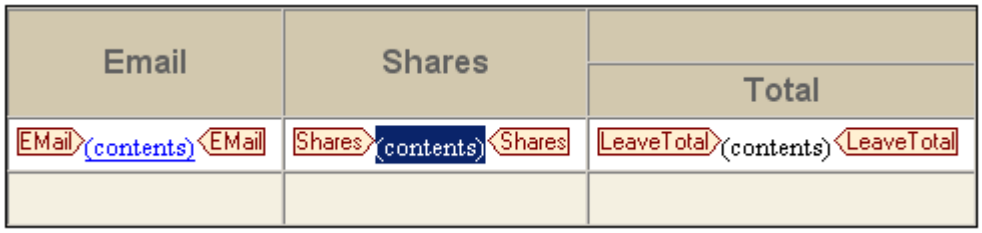
 © 2005 Altova GmbH



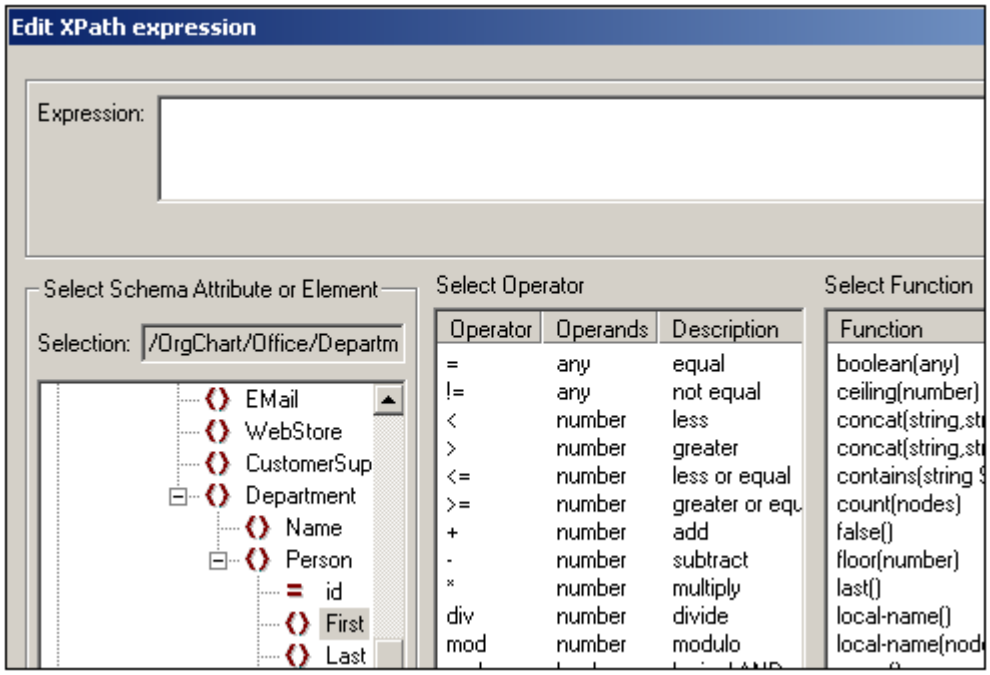
The result is a condition that selects the content of the `//Address_EU/ipo:city` element when the address element is `Address_EU`.

Creating a condition to make shareholders' names bold
In the dynamic table for the `Person` element, you will first create a column for the number of shares each employee holds. Then create a conditional template around the `First` element to display the element's content in bold if the `Share` element is non-empty and greater than 0. Do the following.

1. Place the cursor in the Leave column, and click **Table | Insert Column** or the  icon to insert a column before the Leave column.
2. Type in the column header Shares, join the two header cells of the column, and format the header text.
3. Drag the element `Shares` from the schema tree into the Shares column body. Create the element as content.
4. Set font-size of the `Shares` element content to 10pt. The column should look like this:



4. Check the display of the Shares column in the Authentic and HTML Previews.
5. Now right-click the `(contents)` placeholder of the `First` element, and select **Insert | Condition...**. This pops up the Edit XPath expression dialog:



Notice that in the Select Schema Attribute or Element window, the `First` element is highlighted in gray. This indicates that for the XPath expression you are creating, the context node is the `First` element.

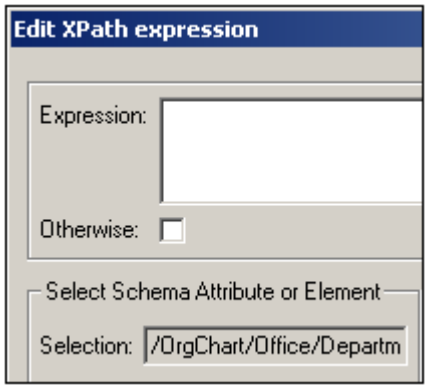
6. Double-click the `Shares` element to insert it into the Expression field. Then select the `>` operator, and type in the value 0. Your XPath expression should be: `../Shares > 0`. Click OK to create the condition.
7. Make the `(contents)` placeholder bold. This means that if the condition is evaluated as true (i.e. if the content of `Shares` (which is a numeric field) is a number greater than 0), then the content of `First` will be displayed in bold. The `First` element in the Design Document should now look like this:



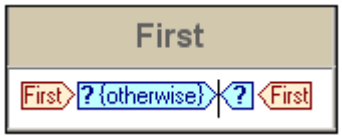
Check the Authentic Editor View. Notice that Frank Further has 0 shares, and his `First` name is not displayed in the table. This is because you have defined processing in the event that the condition is true, but no processing if the condition is false (as is the case with Frank Further since he has 0 shares). You therefore have to define a condition to process the element in the event that the condition is false.

8. To process the element's content when the above condition is false, you must define a condition that specifies all the other eventualities not defined in the above condition. Right-click the condition, and select **Condition | Add New**. The Edit XPath expression dialog appears.

© 2005 Altova GmbH



9. Check the Otherwise box. This condition covers all events not covered by the previously defined condition. The `First` element in the Design Document now looks like this:



Notice that the conditional template is empty. This means that no processing has been defined for it, and, as a result, nothing will be displayed in the output.

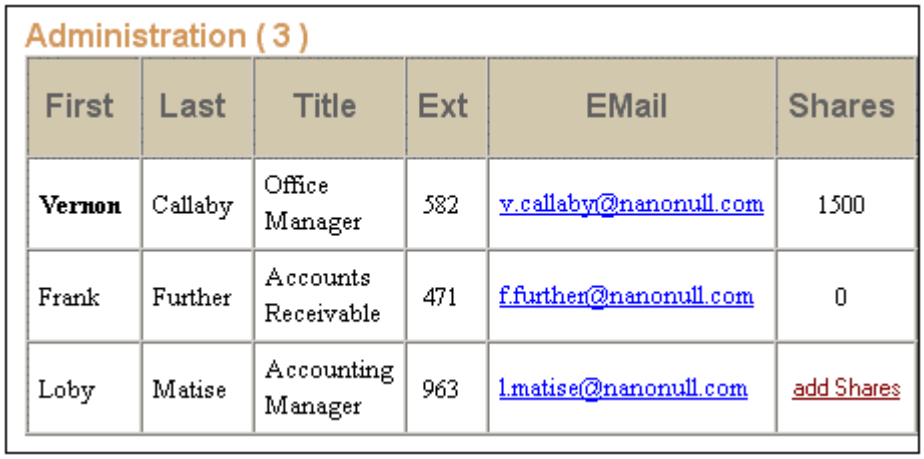
10. Place the cursor between the start and end tags of the condition, right-click, and select **Insert | Contents**. The `(contents)` placeholder is inserted.
11. Select the `(contents)` placeholder, and set font-size to 10pt.

The Authentic and HTML Previews will look like this:

Administration (3)

First	Last	Title	Ext	EMail	Shares
Vernon	Callaby	Office Manager	582	v.callaby@nanonull.com	1500
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	0
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	

The Authentic View will look as below.



Administration (3)

First	Last	Title	Ext	EMail	Shares
Vernon	Callaby	Office Manager	582	v.callaby@nanonull.com	1500
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	0
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	add Shares

Notice in both the above figures, that the `First` element is displayed in normal font if the content of `Shares` is 0 or if no `Shares` element exists for that employee.

Now create the same conditions for the `Last` element so that both `First` and `Last` appear in bold if the employee is a shareholder and in normal if not.

Formatting the table

Now set column widths for each column (by placing the cursor in the respective column, clicking the Table Properties icon, and in the Column tab of the Table Properties dialog entering a value for `width`) as follows: `First`: 10%; `Last`: 12%; `Title`: 16%; `Ext`: 5%; `EMail`: 23%; `Shares`: 10%; `LeaveTotal`: 8%; `LeaveUsed`: 8%; `LeaveLeft`: 8%.

 © 2005 Altova GmbH

3.11 Creating the Table Footer

A table footer can contain both static and dynamic content, and can therefore be used to display summaries of data in the table. You can also insert, append and delete footer rows, as well as split and join cells, thus allowing you considerable flexibility in designing footers.

In this section, we will create the following summaries for the `//Department/Person` dynamic table:

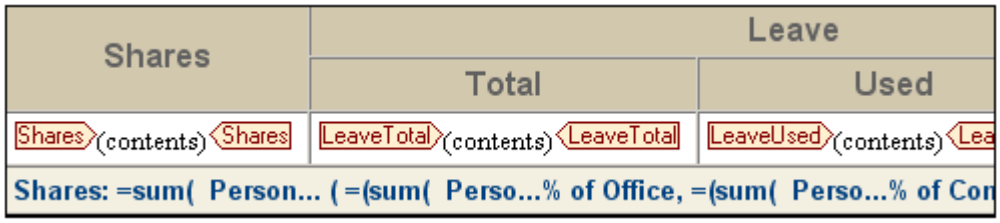
- Number of employees in the department, and this number as a percentage of Office employees and Company employees.
- Number of shares held by Department members, and this number as a percentage of shares held by Office shareholders and Company shareholders.
- A list of all non-shareholders in the department.

Before starting, delete the summary auto-calculation in the footer of the Extensions column, which sums up the telephone extension numbers of employees. This summary was created when the table was created because you checked the Summary for Numeric Fields checkbox and because the `PhoneExt` element is defined as an integer datatype. To delete the summary auto-calculation, click it, and press Delete.

Note that, when the cursor is in the footer, the context node is `//Department`.

Summarizing the shares in departments

1. Place the cursor in the Shares column footer, and click  (or **Table | Join Cell Right**) thrice in order to combine these cells into one cell.
2. Type in "Shares: ".
3. Insert an Auto-Calculation to sum the number of shares held by department employees. The XPath expression will be: `sum(Person/Shares)`.
4. Insert an Auto-Calculation to sum the number of shares held by department employees as a percentage of shares held by all shareholders in that office. The XPath expression will be: `round((sum(Person/Shares)) div (sum(../Department/Person/Shares)) * 100)`. The `round()` function is used to round off the result of the Auto-Calculation.
5. Insert an Auto-Calculation to sum the number of shares held by department employees as a percentage of shares held by all shareholders in the whole company. The XPath expression will be: `round((sum(Person/Shares)) div (sum(../../Office/Department/Person/Shares)) * 100)`.
6. Insert text so that your output reads: `Shares: X ( Y% of Office; Z% of Company )`, where `X`, `Y`, and `Z` are the Auto-Calculations you have created.
7. Format the entire footer text suitably. The Design Document will look like this:



The Authentic and HTML Previews will look like this:

Shares	Leave		
	Total	Used	Left
1500	25	4	21
0	22	2	20
	25	7	18
Shares: 1500 (12% of Office, 4% of Company)			

Summarizing the employees in departments

1. Place the cursor in the footer of the `First` column, and click  (or **Table | Join Cell Right**) four times in order to combine these cells into one cell.
2. Type in "Employees: ".
3. Insert an Auto-Calculation to count the number of department employees. The XPath expression will be: `count(Person)`.
4. Insert an Auto-Calculation to count the number of department employees as a percentage of employees in that office. The XPath expression will be: `round((count(Person)) div (count(../Department/ Person)) * 100)`. The `round()` function is used to round off the result of the Auto-Calculation.
5. Insert an Auto-Calculation to count the number of department employees as a percentage of employees in the whole company. The XPath expression will be: `round((count(Person)) div (count(../../Office/Department/ Person)) * 100)`.
6. Insert text so that your output reads: `Employees: X ( Y% of Office; Z% of Company )`, where `X`, `Y`, and `Z` are the Auto-Calculations you have created.
7. Format the entire footer text suitably. The Authentic and HTML Previews will look like this:

Administration

First	Last	Title
Vernon	**Callaby**	Office Manager
Frank	Further	Accounts Receivable
Loby	Matise	Accounting Manager
Employees: 3 (20% of Office, 8% of Company)		

Listing the non-shareholders by department

1. Place the cursor in the Employees column of the footer, and click  (or **Table | Append Row**). The appended footer row has two cells just like the first row.
2. Place the cursor in the first cell, and click  (or **Table | Join Cell Right**) to join the two cells.
3. Type in "Non-Shareholders: ".
4. Drag the `Person` element into the footer as shown below:



5. Insert a Condition after the start tag of the `Person` element. The XPath expression should select those `Person` elements (the current element) that have a `Share` value of less than or equal to 0, or no `Share` element. The XPath will be: `Shares <= 0 or not (Shares)`. This part of the footer in the Design Document will look like this:



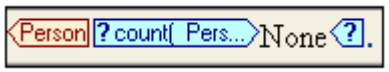
6. Delete the `(contents)` placeholder of the `Person` element.
7. Drag the `First` element into the Design Window, and drop it between the tags of the condition you created in the previous step, and create it as contents.
8. Do the same for the `Last` element. This part of the footer row will now look like this:



9. We now want to put commas after each non-shareholder except the last in the list. To do this, place the cursor after the end tag of the `Last` element, and insert a condition. The XPath should be: `following-sibling::Person[Shares<=0 or not(Shares)]`. The XPath location axis is `following-sibling`, which checks the following siblings of the current `Person` element. The condition searches for following-sibling `Person` elements which contain either `Share` child elements with values equal to or less than `0`, or which contain no `Shares` child element.
10. Insert a comma and a space. This puts a comma and space in the output if the condition is evaluated as true. If not evaluated as true, the conditional template is skipped. Then put a period after the `Person` end tag. This causes a period to be inserted after all the `Person` elements have been looped through. The Design Window of this part of the footer will look like this:



11. You still have to make allowance for the event that there are no non-shareholders in a department (i.e. all department employees are shareholders). To cover this eventuality, place the cursor before the period, and insert a condition. The XPath should be: `count(Person) = count(Person[Shares>0])`. Insert the text "None" between the tags of the condition, as shown below.



12. Format the entire footer row text suitably.
13. Save the StyleVision Power Stylesheet, and refresh the Authentic View of the Working XML File.

The Authentic View of the dynamic table with the footer modifications will look like this:

Administration

First	Last	Title	Ext	EMail	Shares	Leave		
						Total	Used	Left
Vernon	**Callaby**	Office Manager	581	v.callaby@nanonull.com	1500	25	4	21
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	0	22	2	20
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	add Shares	25	7	18
Employees: 3 (20% of Office, 9% of Company)					**Shares: 1500 (13% of Office, 6% of Company)**			
Non-Shareholders: Frank Further, Loby Matise.								

© 2005 Altova GmbH

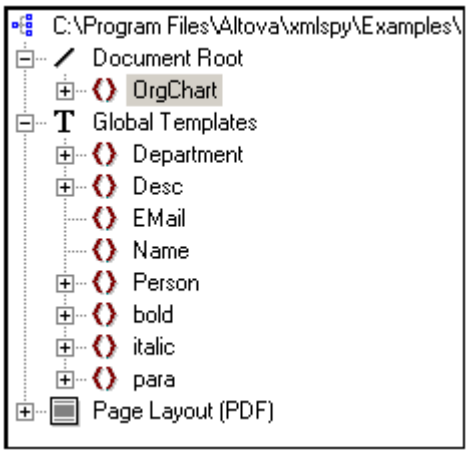
3.12 Text formatting with global templates and text state icons

The schema `NanonullOrg.xsd` contains two **elements** that are intended for applying text styling. These two elements are `bold` and `italic`; they are child elements of `para` and are recursive with respect to themselves and each other, i.e. they can occur multiple times within themselves and each other.

The `bold` and `italic` elements can occur at any location and at multiple locations within a `para` element. Since their location cannot be precisely fixed in the Design Document, the formatting for these two elements must be specified in a global template.

Creating a global template for the `bold` element

1. In the Schema Window, expand the Global Templates entry. A tree showing all the elements for which global templates are allowed is displayed.



2. Select the `bold` element. The `bold` element in the Global Templates tree is marked with a "+" to indicate that a global template has been created. In the Design Window, the view changes to show the `bold` element, which has a `(contents)` placeholder (see below).



3. Note that the entire element is selected. Keeping the element selected, in the Text Style Window, select the Text tab, and set `font-weight` to bold. The global template is now created with bold formatting. Alternatively, you could select the `(contents)` placeholder and assign the bold formatting.



4. Now check the HTML Preview. All text in `para` elements that is marked up with the `bold` element is now displayed in bold. (Such text was earlier not displayed bold because there was no formatting defined for the `bold` element.)

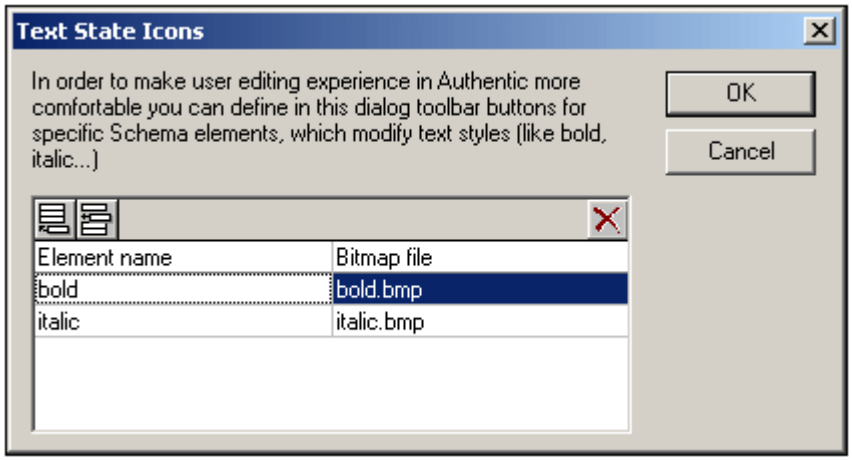
Creating a Text State Icon

A Text State Icon is a toolbar icon in Authentic View (i.e. in Altova's XMLSpy 2005, Authentic 2005 Desktop Edition, and Authentic View products) that is used to add formatting to text in

Authentic View. A text state icon is created, in StyleVision, for **a particular element** and it appears in the Authentic View toolbar in the products listed above (not in Authentic Preview of StyleVision). The formatting for that element must be defined as a global template, and it is this formatting that is applied when the Authentic View user clicks the text state icon.

Create a text state icon for the `bold` element as follows:

1. Click **Authentic | Text State Icons...**. The Text State Icons dialog appears:


Text State Icons
In order to make user editing experience in Authentic more comfortable you can define in this dialog toolbar buttons for specific Schema elements, which modify text styles (like bold, italic...)
OK
Cancel
Element name
Bitmap file
bold
bold.bmp
italic
italic.bmp

2. Enter `bold` as the element name for which the text state icon is being created.
3. Enter `bold.bmp` as the name of the bitmap file in the `sps\Picts` folder in your application folder. This bitmap will be used as the toolbar icon in Authentic View. It is already present in the `sps\Picts` folder, so you do not have to create the bitmap.
4. Now append a row, and add the `italic` element and `italic.bmp` as the bitmap file for the italic icon (also present in the `sps\Picts` folder).
5. Save the StyleVision Power Stylesheet.
6. The toolbar icons can be viewed in Authentic View in XMLSpy 2005, Authentic 2005 Desktop Edition, or Authentic 2005 Browser Edition. They cannot be viewed in Authentic Preview in StyleVision. So, in order to see the toolbar icons for `bold` and `italic` that you have created, you must open `NanonullOrg.xml` in Authentic View in one of the Altova products listed above. If the file is already open in one of these products, close it, and open it again to see the toolbar icons.



   The text state icons are grayed out when the cursor is not inside a `para` element. They will be enabled when you place the cursor inside a `para` element (because the `bold` and `italic` elements are child elements of `para`).
7. In Authentic View, now mark some text in a `para` element, and click the bold icon. The text turns bold.
8. Now mark some text, and click the italic icon. Nothing happens. This is because you have not yet defined a global template for the `italic` element in your StyleVision Power Stylesheet.
9. In the StyleVision Power Stylesheet, create a global template for the `italic` element, and assign it an italic style (in the Text Style Window).
10. Save the StyleVision Power Stylesheet. In Authentic View, refresh the view by switching to another view before switching back again. All text that has been marked up as italic will now be displayed as italic.

 © 2005 Altova GmbH

3.13 Using Additional Validation

The Additional Validation feature allows you to check data on the basis of XPath expressions and to draw the attention of the Authentic View user to values that are outside the range of values specified by the XPath expression. When an invalid value is entered or becomes the result of an auto-calculation, then the user is informed about this in two ways:

- On validating the document, a validation error is reported.
- Optionally, the value can be displayed in red and an error message will be displayed in a mouseover.

This feature is useful if you wish to inform users that certain values are not allowed and that appropriate action is required. Additional validation is also useful when the validity check you wish to make is not specified in the schema.

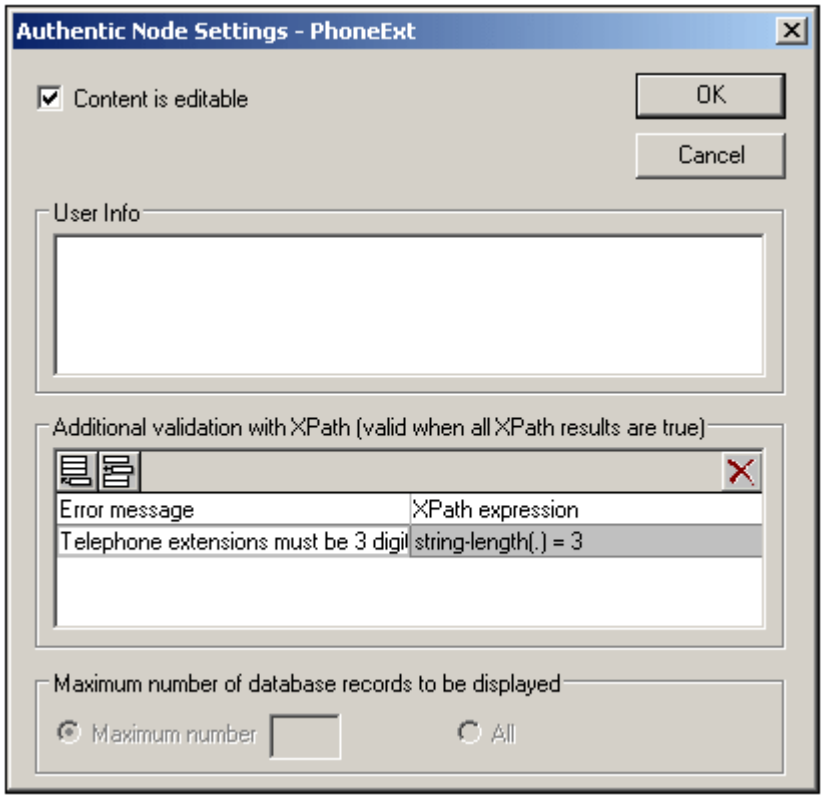
Additional validation of Telephone Extensions
The objective is to validate the content of the `PhoneExt` element, which is an integer datatype. The validation should check whether the extension has three digits or not. If not, the user should be notified.

To build additional validation for the `PhoneExt` element, do the following:

1. Select the `(contents)` placeholder of the `PhoneExt` element.
   - Selecting the `(contents)` placeholder—and not the entire node—is the mechanism to be used in order to display invalid data in red and display an error message when the user mouses over the invalid data. On validation **(F8)** also, an error message will be generated.
   - Selecting the entire node on the other hand only generates an error message on validation.



2. Select **Authentic | Node Settings...** The following dialog appears:



3. In the Additional Validation field, click the Append button.
4. Enter the XPath expression for valid values: `string-length(.)=3`. The . indicates the current node. When the number of characters in the current node (the `PhoneExt` node) is equal to 3, the XPath expression is true and the data is valid. The expression is false when the number of characters in the `PhoneExt` node is any other than 3. If this happens, the node content will be displayed red in Authentic View and a mouseover displays the error message that you key into the Error message box of the Additional Validation field.
5. Now enter the error message in the Error message box of the Additional Validation field, such as: "`Telephone extensions must be three digits long`."
6. You can also key in text for the User Info. The text you key in here is displayed in Authentic View shortly after the cursor is placed over that node.

When the Authentic View user validates the XML document, any invalid data will generate the error message for that node.

© 2005 Altova GmbH

3.14 Completing the stylesheet

To complete the stylesheet, you should check the following:

- Whether Auto-Calculations, Conditional Templates, and Additional Validation statements are correct, especially whether all user situations are covered.
- Whether formatting and layout is as required.
- Whether the Authentic Node Settings for individual nodes are correctly set.

Checking Auto-Calculations, Conditional Templates, and Additional Validation
Two kinds of error are likely here: first, that the XPath expression is incorrect, either in syntax or because it selects the wrong node/nodeset or operates on the selected node/nodeset in an unintended way; second, that some data input scenario has not been covered. You should check all XPaths for correct syntax and correct processing, and then test your StyleVision Power Stylesheet with different data (to check that various data input scenarios are covered).

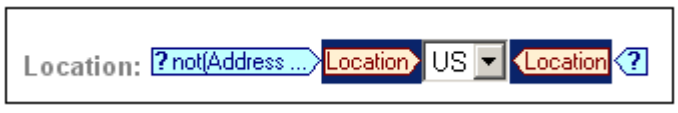
Your StyleVision Power Stylesheet is correct in the first respect, i.e. correctness of XPath expressions, if you have followed this tutorial exactly. There is, however, one significant data input scenario that has not been covered. This is as follows. Suppose the Authentic View user selects a value in the `Location` combo-box, say `US`, and then creates the `Address` element and enters data into the child elements of `Address`. Suppose now that the user (for whatever reason) changes the value in the `Location` combo-box to `EU`. The `add Address_EU` prompt appears, and the user does add the `Address_EU` element. The XML document now contains an `Address` element as well as an `Address_EU` element; this is invalid, since the schema specifies that either of the two, but not both, must be present.

To cover this data input scenario, you can take one of the following measures:

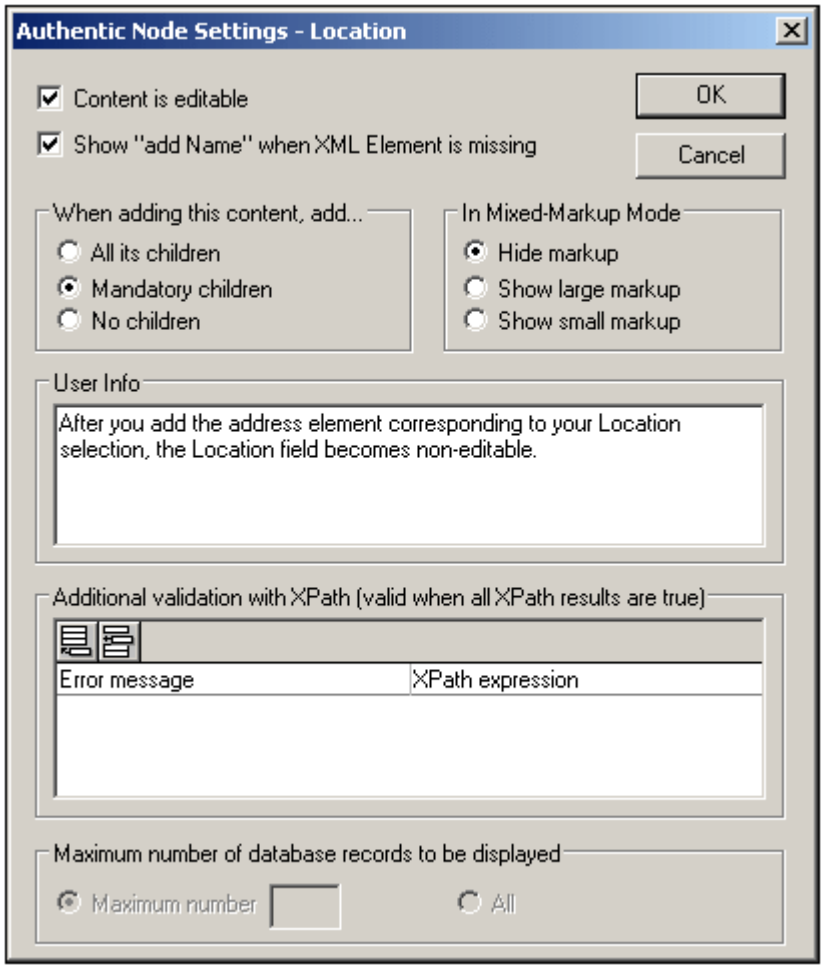
- Inform the Authentic View user about not adding a second address element in a pop-up message that appears on mousing over the `Location` combo-box. You can enter the message in the User Info field of the Authentic Node Settings for the `Location` node. This, however, does not prevent the user from going ahead and doing just what is not supposed to be done.
- Make the `Location` combo-box non-editable once an address element has been created for the Office, and explaining this to the user in a pop-up. This effectively rules out the possibility of two address elements occurring inside an `Office` element.

Additional specifications for how the `Location` combo-box is processed
The objective is to make the `Location` combo-box non-editable after the address element corresponding to the `Location` selection has been created. This prevents the possibility of a second address element being created. Do this as follows:

1. Select the `Location` element, and insert a condition around it with the XPath: `not(Address or Address_EU)`.

   This creates a conditional template for the scenario where neither an `Address` nor an `Address_EU` element exists for that `Office` element.
2. Right-click the `Location` node and select **Authentic Node Settings...** (or select **Authentic | Node Settings...**).


Authentic Node Settings - Location
Content is editable
Show "add Name" when XML Element is missing
OK
Cancel
When adding this content, add...
All its children
Mandatory children
No children
In Mixed-Markup Mode
Hide markup
Show large markup
Show small markup
User Info
After you add the address element corresponding to your Location selection, the Location field becomes non-editable.
Additional validation with XPath (valid when all XPath results are true)
Error message
XPath expression
Maximum number of database records to be displayed
Maximum number
All

In the dialog, leave the default settings unchanged. The most important of these in this context is the "Content is editable" checkbox. Remember that these settings are for the condition where neither `Address` nor `Address_EU` element exist; you do want the user to be able to add content to the `Location` element.

3. Add text in the User Info field as shown in the figure above, and click OK.
4. Right-click the condition and select Add Copy.
5. For the new conditional template, change the XPath to: `Address`. This is the conditional template to execute when an `Address` element exists.
6. Select the `Location` node in this conditional template, and in the Node Settings dialog uncheck the "Content is editable" checkbox and enter a suitable text message in the User Info field, such as: "To edit this field, the `Address` element of this `Office` element will have to be deleted using a Text View of the document."
7. Repeat Steps 4, 5 and 6 to create a third conditional template for cases where an `Address_EU` element exists.

Checking the document formatting

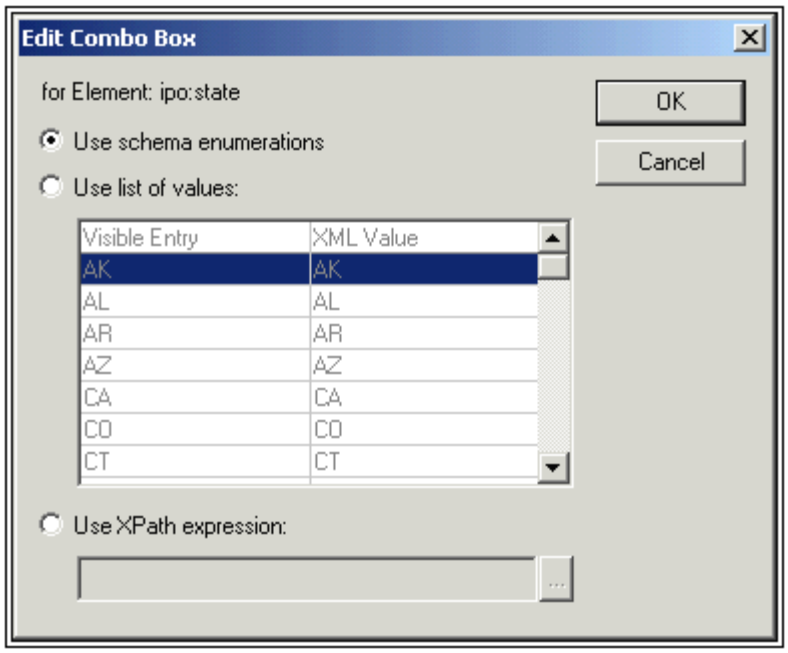
Compare the Authentic View and HTML Preview, and make any desired formatting changes, such as adding or deleting line breaks, changing text color and background color, and so on.

In the static table that contains the US address fields, the last row contains two elements: `ipo:state` and `ipo:zip`. In order to make the input areas for the two elements easily recognizable to the Authentic View user, change the `ipo:state` and `ipo:zip` nodes to a Combo box and an Input Field, respectively.

 © 2005 Altova GmbH

To change the `ipo:state` element to a combo box, do the following:

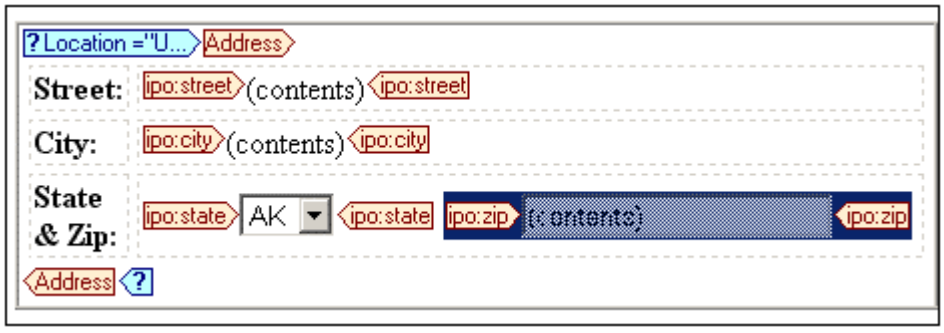
1. Right-click the element in the Design Document and select **Change to | Combo Box...**. The following dialog pops up:



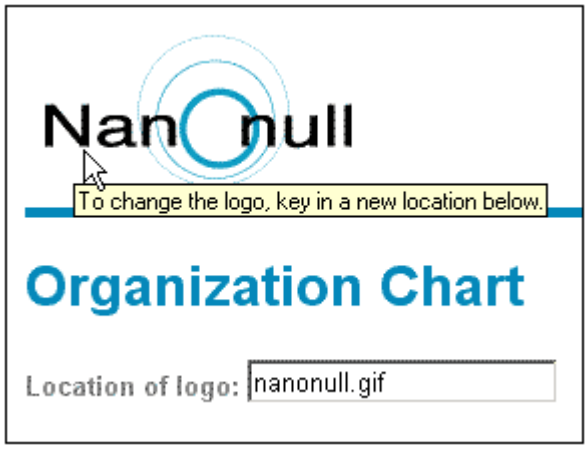
2. Check the Use Schema Enumerations option, so that the values enumerated in the schema become the options available in the combo box. The Visible Entries will be the same as the XML Values that have been enumerated in the schema.
3. Click OK to finish creating the combo box.

To change the `ipo:zip` element to an input field, right-click the `ipo:zip` element and select **Change to | Input Field**.

The Design Window, after these two changes have been made, should now look like this:



Checking the Node Settings
You can make detailed and specific settings for each node in your document (right-click the node and click **Authentic Node Settings...** (or select **Authentic | Node Settings...**)). For example, you might want to add User Info for some nodes; Authentic View users will see this information when they mouseover the node. The following screenshot shows how User Info is displayed in Authentic View.



The pop-up message that appears on mousing over the logo is the text keyed in as the User Info for this node.

Additional modifications
Try the following modification yourself. In the Office Summary Line just below the address of each office, the number of departments and employees are listed, as shown here:



Modify the line so that the singular (department / employee) appears when the corresponding value is one, and the plural (departments / employees) appears when the corresponding value is zero or more than one. (This modification is included in `NanonullOrg.sps`, should you need to look it up.)

 © 2005 Altova GmbH

Chapter 4

How To Use

4 How To Use

This section describes how to use the more complex features of StyleVision. For each of the features considered, there is an explanation of the feature's function followed by a description of its usage.

The following features are discussed:

- XSLT and XPath versions
- Namespaces
- Global Templates
- Data-entry devices
- Lists
- Tables
- Auto-Calculations
- Conditional templates
- Date Picker
- Input Formatting
- Parameters
- Unparsed Entity URIs
- StyleVision from the command line
- Authentic View features

© 2005 Altova GmbH

4.1 XSLT and XPath versions

StyleVision has separate XSLT 1.0 and XSLT 2.0 engines. The interface supports correspondingly different functionality depending on the XSLT version selected for a particular StyleVision Power Stylesheet. For each StyleVision Power Stylesheet you create, you must select the XSLT version to which that StyleVision Power Stylesheet must conform. You do this by clicking the appropriate toolbar icon:  or  . The selection you make determines two things:

- Which of the two XSLT engines in StyleVision is used for transformations.
- What XSLT functionality is displayed in the interface.

Transformations are used in StyleVision (i) to generate previews in the interface; and (ii) to generate and save output files (HTML) from within the interface and from the command line. The XSLT engine used for transformations (XSLT 1.0 or XSLT 2.0) is that for the version selected in the StyleVision Power Stylesheet.

The functionality appropriate for each XSLT version relates mostly to the use of the correct XPath version (XPath 1.0 for XSLT 1.0 and XPath 2.0 for XSLT 2.0). XPath is widely used in StyleVision—most commonly in features such as Auto-Calculations, conditions, and dynamic data selection—and there are a number of interface mechanisms that require, and help you build, XPath expressions. The functionality of the correct XPath version is automatically made available in the interface.

4.2 Namespaces

Namespaces with DTDs and XML Schemas
When using namespaces, it is important to know that:

- DTDs **are not** namespace aware
- XML Schemas **are** namespace aware

This means that in an XML document, according to the Namespace Recommendation, an element such as `altova:title` is the same as the element `title` if both belong to the same namespace. The prefix `altova:` is not a part of the element name, but is a reference to a namespace. However, since DTDs are not namespace aware, they recognize `altova:` as part of the element name. Consequently, a DTD would see `altova:title` and `title` as two different elements.

StyleVision does not support the use of namespaces in DTDs. If you wish to use namespaces with your documents, we recommend that you use XML Schemas.

Problems of namespaces in DTDs
If you wish to use a DTD to validate an XML document that uses namespaces, you would have to declare `xmlns` (with an optional prefix) as an attribute of the relevant element. For example:

```
<!ATTLIST userman xmlns:altova CDATA #FIXED
   "http://www.altova.com">
```

Further, elements must be declared in the DTD with the namespace prefix as part of the element name, as for example:

```
<!ELEMENT altova:title (#PCDATA)>
```

A valid XML document based on such a DTD would look something like this:

```
<altova:userman version="1.0" xmlns:altova="http://www.altova.com">
   <altova:title>Some Title</altova:title>
   ...
</altova:userman>
```

In such a DTD, you would have to define all elements and attributes with the namespace prefix as part of the name. This means that when you create an XML instance document you are committed to the prefix you use in the DTD. If you subsequently wish to change the prefix in the XML document then all the element and attribute declarations in your DTD would have to be modified. You could gain a little flexibility by using parameter entities, but, clearly, using a DTD to validate an XML document with namespaces is not an ideal strategy .

If you wish to use namespaces, it is best to base your StyleVision Power Stylesheet on an XML Schema.

In what follows the focus is on XML Schemas. The topics covered are:

- A general overview of namespaces in StyleVision
- How StyleVision handles a schema with a single namespace
- How StyleVision handles a schema that uses multiple namespaces

© 2005 Altova GmbH

4.2.1 Namespaces and StyleVision

Target namespaces
If you wish to use a namespace with your XML document instance, a target namespace must be defined in the XML Schema. This is done with the `targetNamespace` attribute of the `schema` element. If the `targetNamespace` attribute is specified, then a namespace for the target namespace must also be defined for the schema.

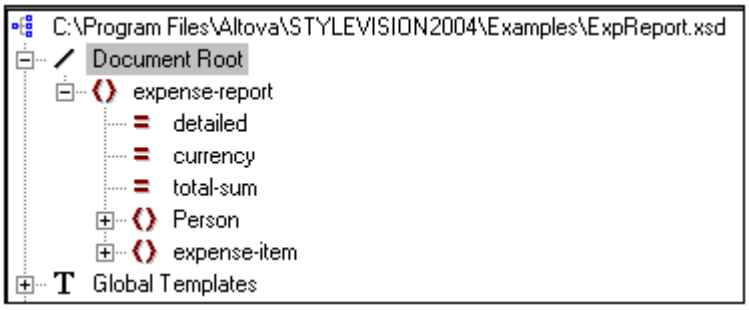
```
<xsd:schema targetNamespace="http://www.xmlspy.com/schemas/orgchart"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:org="http://www.xmlspy.com/schemas/orgchart"
   elementFormDefault="qualified"
   attributeFormDefault="qualified">
```

The handling of namespace prefixes in StyleVision is therefore dependent on (i) whether a target namespace is defined, and (ii) whether the target namespace has been defined with a prefix (as in the example above) or as the default namespace. The handling of namespace prefixes in StyleVision is explained in detail in Schemas with a single namespace and Schemas with multiple namespaces.

Schemas with no target namespace
If you do not wish to use a namespace with your document instances, then the `targetNamespace` attribute is omitted in the schema. In these cases the document instance does not belong to any namespace, and the schema tree in StyleVision reflects this in that elements and attributes have no prefix.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   attributeFormDefault="qualified">
```

The schema tree in StyleVision for such a schema will have no namespace prefixes.



Qualifying elements and attributes
An element or attribute in the XML instance document is said to be qualified when its namespace prefix is exposed. Whether an element or attribute is qualified depends on the values of the `elementFormDefault` and `attributeFormDefault` attributes of the schema element, respectively, and, in the case of elements, whether the element is globally declared or not.

When the `elementFormDefault` and `attributeFormDefault` attributes have a value of `qualified`, then elements and attributes in the XML instance document must have their prefix exposed. When the value is `unqualified`, then only globally declared elements must have their prefix exposed. The effect of the values of the `elementFormDefault` and `attributeFormDefault` attributes on the handling of namespace prefixes in StyleVision is detailed in Schemas with a single namespace and Schemas with multiple namespaces.

Note: The default values of the `elementFormDefault` and `attributeFormDefault` attributes are both `unqualified`. As a result, if these attributes are not specified, then they are

taken to be present with a value of `unqualified`.

Importing schema modules
A schema would typically belong to a single namespace. Schema modules that belong to different namespaces can be combined by using the `xsd:import` element. Since each schema has its own `elementFormDefault` and `attributeFormDefault` attributes, mixed values across combined schemas can produce complex namespace prefixing in StyleVision. These rules are given in Schemas with multiple namespaces.

Namespace binding
When the XML instance document is created, it will be in the target namespace. The namespace declaration in the XML document can bind the target namespace to any prefix. It does not necessarily have to be the same binding as that used in the schema or StyleVision Power Stylesheet. What is important is that the namespace URI referenced by the namespaces in each document be the same. So, even though a namespace in the schema has a prefix of `doc:` in the schema, the StyleVision Power Stylesheet, and the generated XSLT stylesheets for HTML, and the same namespace has a prefix of `org:` in the XML instance document, the XML instance file is valid and will transform correctly with the stylesheets.

If you are creating an StyleVision Power Stylesheet for use with Authentic View, we recommend that you create a Template XML File that uses the same prefixes as those used in the schema and StyleVision Power Stylesheet. This would ensure that an XML file based on your StyleVision Power Stylesheet will have the same namespace prefixes as your schema and StyleVision Power Stylesheet.

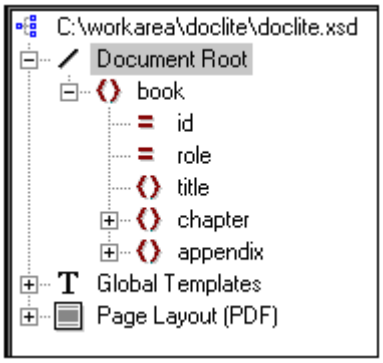
© 2005 Altova GmbH

4.2.2 Schemas with a single namespace

Target namespace is default namespace of the schema and all nodes are qualified
When a target namespace is specified in the schema, then the XML instance document belongs to this namespace. In the schema fragment shown below, a target namespace is specified, and the same namespace is defined as the default namespace of the schema document (that is, the namespace is defined without a prefix). Note also that `elementFormDefault` and `attributeFormDefault` are set to `qualified`.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.altova.com/workarea/doclite"
   xmlns="http://www.altova.com/workarea/doclite"
   elementFormDefault="qualified"
   attributeFormDefault="qualified">
```

The schema tree in StyleVision for such a schema—which specifies that elements must be qualified—will have no namespace prefixes. This is because the target namespace is the default namespace (which has no prefix). The same will also be the case if `attributeFormDefault` is set to `unqualified`. For the handling of namespace prefixing when `elementFormDefault` is set to `unqualified`, see below.



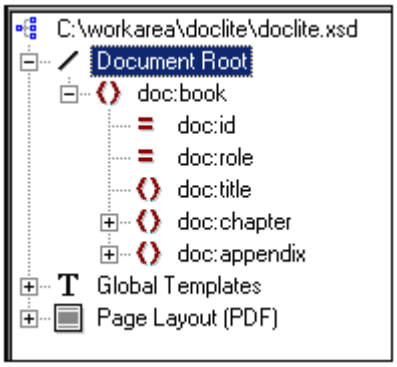
Note:

- If both `elementFormDefault` and `attributeFormDefault` are set to `qualified`, attributes will not be prefixed.
- In the XML instance document, however, qualified attributes must explicitly specify a namespace prefix even if they are in the default namespace. This is because the specification does not provide a mechanism for defaulting the namespaces of attributes. So, if attributes have to be qualified, you should define a prefix for the namespace instead of using a default namespace.

Target namespace is defined with a prefix in the schema and all nodes are qualified
In this schema example, a target namespace is specified, and the same namespace is defined with a prefix `doc:` in the schema document. Note also that `elementFormDefault` and `attributeFormDefault` are set to `qualified`.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.altova.com/workarea/doclite"
   xmlns:doc="http://www.altova.com/workarea/doclite"
   elementFormDefault="qualified"
   attributeFormDefault="qualified">
```

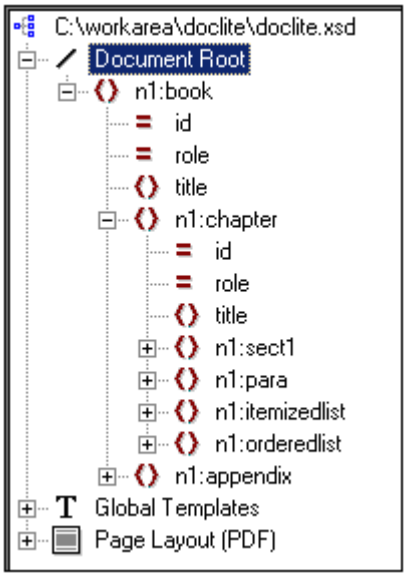
The schema tree in StyleVision for this schema will have all elements and attributes prefixed with `doc:` since the `elementFormDefault` and `attributeFormDefault` are set to `qualified`. If `attributeFormDefault` is set to `unqualified`., then the prefix will be hidden for attributes. For the handling of namespace prefixing when `elementFormDefault` is set to `unqualified`, see below.



Target namespace is default namespace of the schema and all nodes are unqualified
When the target namespace is the default namespace of the schema document and `elementFormDefault` is set to unqualified, then the prefix is exposed for global elements but is hidden for other elements. Note that, in this example, `elementFormDefault` and `attributeFormDefault` are set to `unqualified`.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.altova.com/workarea/doclite"
   xmlns="http://www.altova.com/workarea/doclite"
   elementFormDefault="unqualified"
   attributeFormDefault="unqualified">
```

Since the target namespace is defined as the default namespace of the schema—that is, no prefix is defined for it—StyleVision assigns an arbitrary namespace prefix `n1:` to the namespace (required because the prefix has to be exposed for global elements). This prefix is exposed on the global elements (all elements except `title`). If `attributeFormDefault` is set to `unqualified`, as is the case above, then the namespace prefix (`n1:`) is hidden on all attributes. If `attributeFormDefault` is set to `qualified`, then the namespace prefix (`n1:`) is exposed on attributes

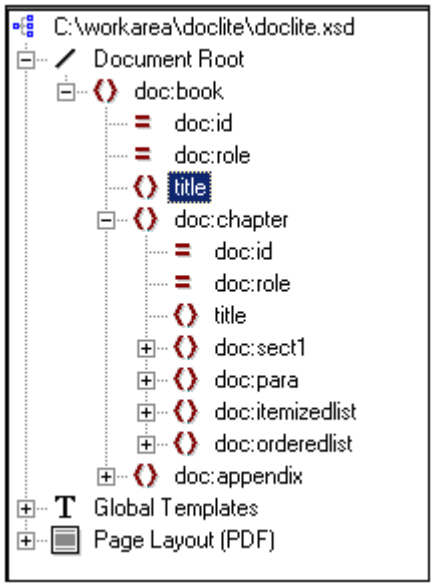


Target namespace is defined with a prefix in the schema and elements are unqualified
When `elementFormDefault` is set to unqualified and the target namespace is bound to a prefix (in this case `doc:`), then the prefix is exposed for global elements but is hidden for other elements.

 © 2005 Altova GmbH

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.altova.com/workarea/doclite"
   xmlns:doc="http://www.altova.com/workarea/doclite"
   elementFormDefault="unqualified"
   attributeFormDefault="qualified">
```

Notice that `attributeFormDefault` is set to `qualified`. As a result, all global elements and all attributes have the `doc:` prefix. Only the `title` elements, which are not global (that is, they are declared locally) have their prefixes hidden.

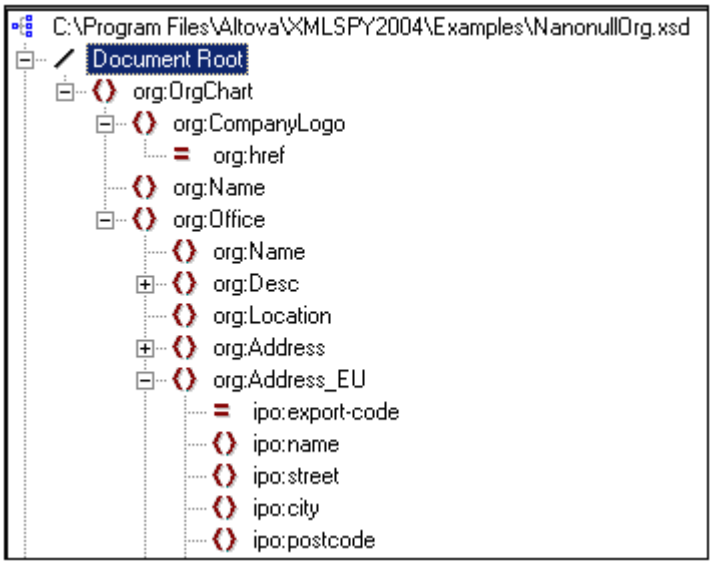


4.2.3 Schemas with multiple namespaces

Schema modules that belong to different namespaces can be combined by using the `xsd:import` element.

```
<xsd:schema targetNamespace="http://www.xmlspy.com/schemas/orgchart"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:org="http://www.xmlspy.com/schemas/orgchart"
   xmlns:ipo="http://www.altova.com/IPO"
   elementFormDefault="qualified"
   attributeFormDefault="qualified">
   <xsd:import namespace="http://www.altova.com/IPO"
schemaLocation="address.xsd"/>
```

If the imported schema also has its `elementFormDefault` and `attributeFormDefault` set to qualified, all namespace prefixes are exposed, as shown below.



Important: The `elementFormDefault` and `attributeFormDefault` attributes apply to the schema they are in. The values of these attributes in an importing schema **do not** override the values of these attributes in an imported schema. Note also that the default value of these two attributes is `unqualified`.

Namespace prefixing with combined schemas
Since each schema has its own `elementFormDefault` and `attributeFormDefault` attributes, mixed values across the various schemas can produce complex namespace prefixing. The rules followed by StyleVision are given in the tables below. The tables below show how StyleVision displays the prefixes of elements and attributes from the main schema and the imported schema, respectively, according to:

- the values of the `elementFormDefault` and `attributeFormDefault` attributes of each; and
- whether the target namespace is set as the default namespace of the main schema or with a defined prefix.

Handling of element namespace prefixes
The following table gives StyleVision's handling of the namespace prefixes of elements. Note that Main Schema column describes the handling of Main Schema elements while the Imported Schema column describes the handling of Imported Schema elements. **EFD** stands for **`elementFormDefault`**, NS for Namespace.

elementFormDefault (EFD)

	Main Schema		Imported Schema	
	Target Namespace has Namespace Prefix	**Target Namespace is Default NS (no prefix)**	**EFD qualified**	**EFD unqualified**
Main Schema EFD qualified	Prefix of all elements exposed.	Prefix of all elements exposed; Target NS prefix is default (no prefix).	Prefix of all elements exposed.	Prefix of global elements exposed.
Main Schema EFD unqualified	Prefix of global elements exposed.	Prefixes of global elements exposed. Target Namespace prefix is `n1:`.	Prefix of all elements exposed.	Prefix of global elements exposed.

Handling of attribute namespace prefixes
The following table gives StyleVision's handling of the namespace prefixes of attributes. The EFD value of the Main Schema does not have an effect on the prefixing of attributes. Note that Main Schema column describes the handling of Main Schema attributes while the Imported Schema column describes the handling of Imported Schema attributes. **AFD** stands for **`attributeFormDefault`**, NS for Namespace.

attributeFormDefault (AFD)

	Main Schema		Imported Schema	
	Target Namespace has Namespace Prefix	**Target Namespace is Default NS (no prefix)**	**AFD qualified**	**AFD unqualified**
Main Schema AFD qualified	Prefix of all attributes exposed.	Prefix of all attributes is `n1:`.	Prefix of all attributes exposed.	Prefix of all attributes hidden.
Main Schema AFD unqualified	Prefix of all attributes hidden.	Prefix of all attributes hidden.	Prefix of all attributes exposed.	Prefix of all attributes hidden.

Note: If the Target Namespace is the default namespace and EFD is qualified, when AFD is qualified, attributes will not be prefixed.

4.3 Global Templates

Function
The Design Document consists of a **Root Template** and one or more optional global templates.

The Root Template defines the structure of the output document. For example, it could specify the following sequence:

- Name of company
- Company address and contact details
- List of products
- Board of directors

An alternative sequence could specify the board of directors before the list of products. Each item in the Root Template is defined by means of a **local template**. What is important to note is that the Root Template determines the structure of the entire output document from beginning to end.

Additionally, **global templates** can be defined for certain elements. If the Root Template invokes the global template of an element at some point during its processing, then the rules of this global template are processed at that point, after which the processor returns to processing the rest of the Root Template. An extreme case of the use of global templates is when the Root Template does not specify any structure but only that templates be applied to children of the Root Element. In such a case, elements are processed in the source XML document order, with global templates being applied if these exist; otherwise **default templates** are used. The default templates output element content. In such a case the structure of the output document follows the structure of the source XML document. This is in direct contrast to cases where the structure of the output document is specified explicitly with the Root Template.

Global templates can be created for individual element types, and they define the processing of elements with that name wherever they occur in the XML document. For example, you could create a global template for the element `para`. The advantage of using a global template for `para` would be that you define in one place the same processing rules for all `para` elements in the XML document. When the XML is processed, all `para` elements (whether they occur in `chapter`, `section`, `appendix` or `footnote`) will be processed according to the rules defined in the global template for `para`. You can also build variations for different `para` elements by using conditional templates within the global template. Thus, you can make a condition so that a `para` in a `footnote` has text that is two points smaller in size than text of other `para` elements.

When to use global templates and local templates
If you use only global templates, the processor will run through the XML document, and for each element that it encounters that has a global template, it will process that element accordingly. So it will follow the XML document structure. This is generally fine if your XML is a text document like an article, because the structure of the output document will generally correspond to the XML document structure.

Since the Root Template specifies the structure of the output document, you should use the Root Template (and local templates within it) if you wish to:

- Produce an output with a structure that is different from the XML document structure (for example output an Appendix before the Chapters)
- Insert overarching external components in the output (for example, a table that includes content from various parts of the XML document)

In some cases, using a global template is the only way to process an element. Global templates can—and, in some cases, must—be used in the following situations:

 © 2005 Altova GmbH

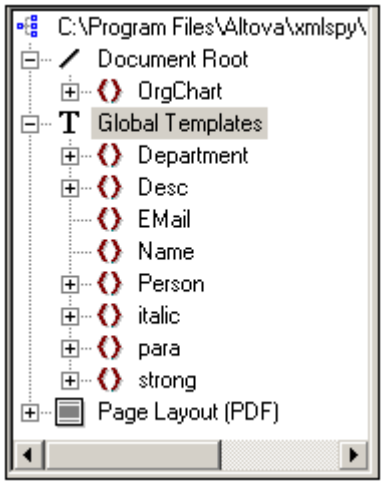
- For elements that can occur at multiple locations in various parts of the document, and where the locations cannot be known to you, the designer of the document, a global template **must** be used. An example of such an element is one marking up italic or bold text. (You cannot know where such elements will occur in the text; therefore, a global template for each such element must be used.)
- If an element can occur recursively, i.e. within itself or a descendant, then a global template **must** be used. An example is a numbered list, an item of which may itself contain a numbered list.
- For an element type that occurs as a child of different parents (for example, `//office/email` and `//person/email`), using a global template ensures that both `//email` elements are processed in the same way—if, that is, you want to process them in the same way. The advantage of using a global template in this kind of scenario is that the processing for several elements of a single element type is defined in a single global template; a change in the global template will therefore apply to all the elements that use this global template.

Very often, it is best to use a combination of both the Root Template (for control of structure) and global templates (for universal application of certain rules). What you use depends entirely on the output you want from your XML. Note that StyleVision allows you to apply global templates within the Root Template. So you can define a default formatting in a global template and then use it or override it in the Root Template.

General mechanism

- A local template defines the processing of one particular element in the Root Template.
- A global template defines the processing of an element type. For example, a global template can be created for the element type `title`, and this global template could then be used for any `title` element in the document (e.g. `//book/title`, `//chapter/title`, `//section/title`, etc).
- You can build variations in the global template by using conditional templates for particular elements (for example, `//book/title` could be given a different text color).
- If an element used in the Root Template has a global template defined for it, either the local template or the global template can be used.
- If an element only has a global template defined, then, in order for the template to be processed and produce output, it must be "reached". It is reached if there are other (local or global) templates that specify processing for each element in the hierarchy, starting from the root element and reaching down to this element.

Elements for which Global Templates are allowed to be created
When a schema is loaded into StyleVision, a single Global Templates entry is created in the Schema Window. This entry contains all the elements for which global template rules are allowed.



Elements that appear in the Global Templates list are:

- All elements named in element declarations in a DTD
- All elements declared as global elements in an XML Schema, i.e. elements declared as children of the `xs:schema` element.

Creating Global Templates
Global templates can be created in either of two ways.

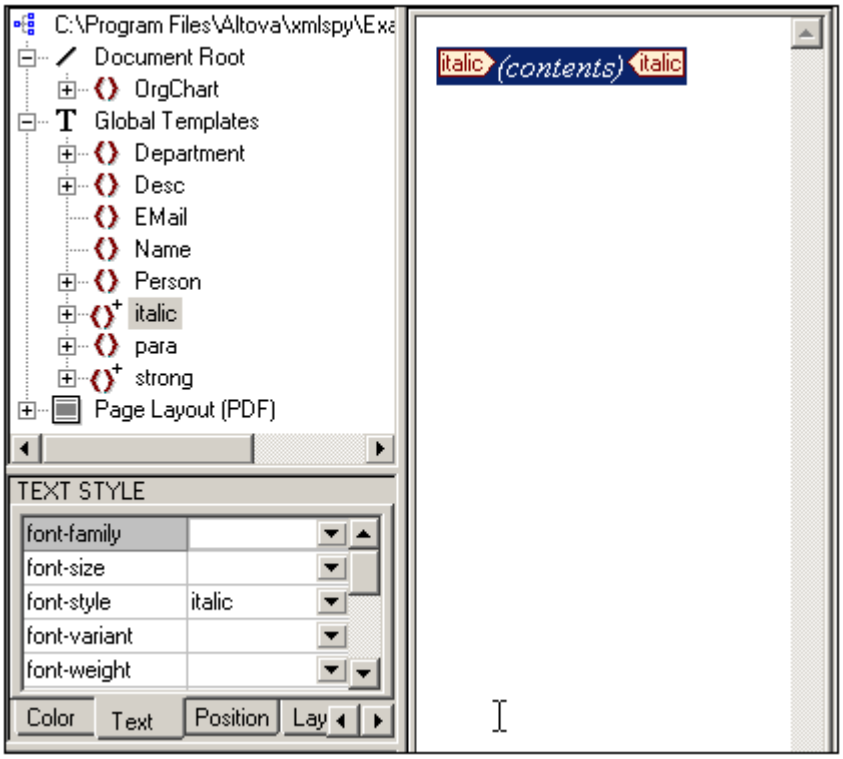
1. By clicking an entry in the Global Templates list and defining properties for it. Clicking an element listed under Global Templates will create an *empty* global template for that element. An empty global template has only the `<apply-templates />` rule specified but no other template rule; that is, it will process child elements but not do anything else (such as format text).
2. By right-clicking an item in the Design window, and selecting Make Global Template. A global template is created for that element type *from the local template* (that is, the global template will be created with the properties of the local template); the local template is deleted; and the global template is used for that element. Note that the Make Global Template command is displayed in the context menu only for those elements that are listed in the Global Templates list.

Whenever a global template is created (empty or not), a "+" sign appears to the left of the element's name in the "Global Templates" list (see figure below). Note that a global template can be created only for the global element itself—not for the descendants that are displayed when the global element is expanded.

After a global template is created, it will not automatically be used for an element in the Design Document—unless it has been created from a local template. The use of the global template for an element in the Design Document must be explicitly specified. This is described in the section "Using Global Templates" below.

Defining properties for Global Templates
To add formatting properties to an empty global template, key in values for the required text properties in the Text Style and/or Block Style windows. In the example below, the global template for the `<italic>` element has been assigned a font-style value of "italic".

 © 2005 Altova GmbH



In the example above, note that only the font-style property has been assigned a value. Other properties, such as font-family, font-size, and font-weight have not been assigned any value. For these properties, the values are inherited from the respective parent of each *instance* of `<italic>`. So if the parent of one instance of `<italic>` were `<para>`, and `<para>` had a font-size value of "11pt", then this instance of `<italic>` would inherit the font-size value of "11pt". If the parent of another instance of `<italic>` were `<email>`, and `<email>` had a font-size value of "10pt", then this instance of `<italic>` would inherit the font-size value of "10pt". Defining the global template for `<italic>` as above has the advantage that the `<italic>` markup changes only the font style (to italic) while the rest of the formatting is exactly the same as the surrounding text, no matter what the surrounding text. For example, an `<italic>` element contained within a `<bold>` element would inherit the font-weight value of "bold", thus producing text in bold italic.

Note: You can use any StyleVision feature in a global template, such tables, conditional templates, and auto-calculations.

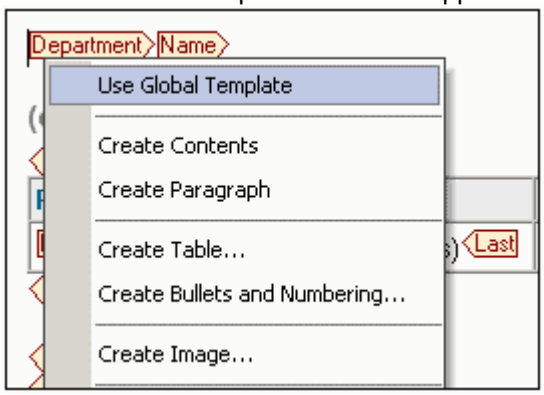
Using Global Templates
Global templates are *created* in either of the two ways described in the section "Creating Global Templates" above. If you wish to use a global template with any element occurrence after it has been created, you have to specify this explicitly.

By default each occurrence of an element uses the local template. So, for example, if from the three occurrences of an `<email>` element: `//company/address/email` and `//employee/office/email` and `//employee/home/email`, you wish to use the global template with the two occurrences: `//employee/office/email` and `//employee/home/email`, then you have to specify that these two occurrences should use the global template.

There are two ways to specify that a given element occurrence should use a Global Template:

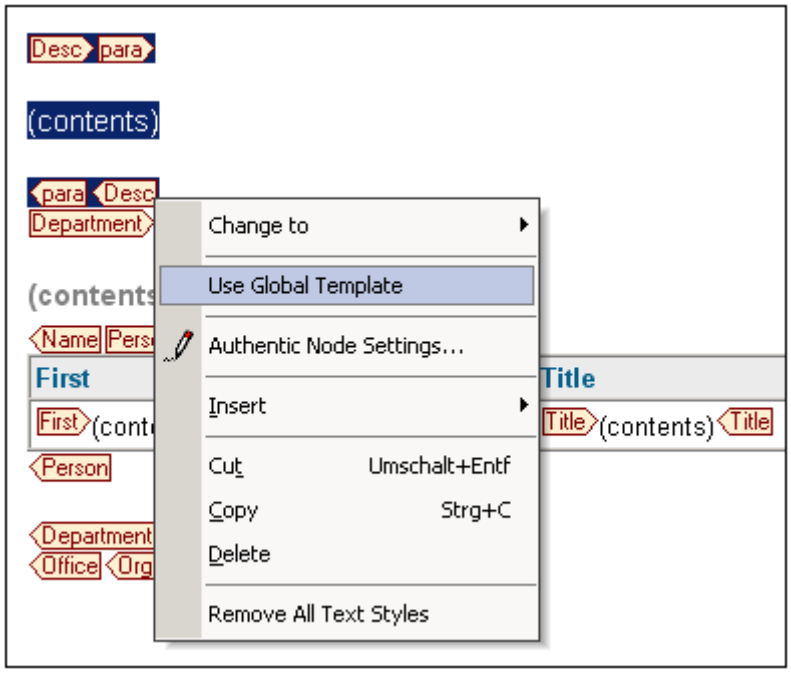
1. If a global template exists for a given element name, and an element with that name is

then dragged from the Schema window, then on dropping the element into the Design window a menu (shown below) pops up. In the figure below, the <Desc> element, for which a global template exists, has been dragged to a location just before the Department tag (location shown by cursor). Since a global template exists for <Desc>, the Use Global Template command appears in the pop-up menu.



Clicking Use Global Template, creates this element occurrence (<Desc>, in the example above) at that location using the global template properties. The element tags will be gray, indicating that the element uses the global template.

2. If the element occurrence already exists, right-click the element in the Design window, and click "Use Global Template". In the figure below, the <Desc> element has been right-clicked. Since a global template exists for <Desc>, the Use Global Template command appears in the context menu.



The local template will be deleted, and the global template will be used. The element tags turn gray, indicating that the element uses the global template.

Creating dynamic tables in global templates

You can also create dynamic tables in global templates. You do this in the same way you create dynamic tables in the Root Template. The important point to note is that, in a global template, a dynamic table can only be created for **descendant elements** of the global template node; it

 © 2005 Altova GmbH

cannot be created for the global template node itself. For example, if you wish to create a dynamic table for the element `authors` within a global template, then this dynamic table must be created within the global template of the parent element of `authors`. It cannot be created within the global template of the `authors` element.

Copying Global Template properties to a Local Template
When an element uses a global template, no local template exists. You can, however, create a local template from the global template. This is useful when you have a global template for an element and wish, for example, to change the formatting for a majority of element occurrences but retain the current global template properties for a few element occurrences. In such a case, you would copy the global template to local templates wherever required, and then make the change in the global template.

To copy a global template to a local template, in the Design window, right-click the element that uses the global template (the element tags will be gray), and click "Copy Global Template Locally". The global template is copied as the local template for the element, and the newly created local template is applied. The element tags turn from gray to beige, indicating that the local template is being used. The global template remains unchanged; it can be used for other occurrences of that element type.

Editing a Global Template
To change the formatting properties of a global template, do the following:

1. Select the Design window tab.
2. In the Schema Window, from the list of Global Templates, click the global template you wish to edit.
3. Enter the required values in the Text Style and/or Block Style windows.

The changes in the global template will be applied to all the element occurrences that use this particular global template.

Removing a Global Template
To remove a global template, do the following:

1. Select the Design window tab.
2. In the Schema Window, from the list of Global Templates, right-click the global template you wish to remove.
3. Select "Remove Global Template".

This removes the global template.

Caution: Use this command with care since it not only removes the global template but also removes *any element type occurrence that is currently using this global template*. If you wish to not have these element occurrences deleted, then ensure that you switch these occurrences to use a local template before you remove the global template.

For the effect of global templates on `rest-of-contents`, see Rest-of-Contents.

4.4 Rest-of-Contents

The `rest-of-contents` placeholder applies templates to all the remaining child elements of the element for which the template has been created. As an example consider the following:

- An element `parent` has 10 child elements, `child1` to `child4`.
- In the template for element `parent`, some processing has been explicitly defined for the `child1` and `child4` child elements.

This results in only the `child1` and `child4` child elements being processed. Now, if the `rest-of-contents` placeholder is inserted within the template for `parent`, then, not only will `child1` and `child4` be processed using the explicitly defined processing rules in the template. Additionally, templates will be applied for the `child2` and `child3` child elements. If global templates for these are defined then the global templates will be used. If these child nodes have built-in default templates (such as for element, attribute, and text nodes), then the built-in templates are applied.

Important: It is important to note what nodes are selected for `rest-of-contents`.

- As described with the example above, all child element nodes and child text nodes are selected by the `rest-of-contents` placeholder.
- "Child" attribute nodes are not selected.
- If a global template of a child element is used in the parent template, then the child element does not count as having been used locally. As a result, the `rest-of-contents` placeholder will also select such child elements. However, if a global template of a child element is "copied locally", then this usage counts as local usage, and the child element will not be selected by the `rest-of-contents` placeholder.

Also see: Inserting Rest-of-Contents and Global Templates.

© 2005 Altova GmbH

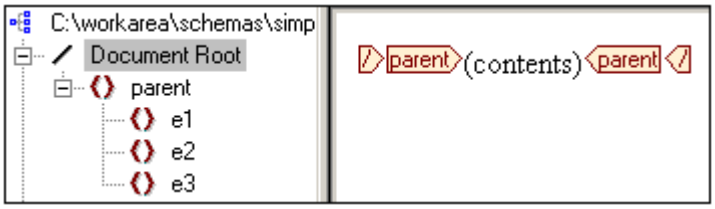
4.5 Adding Elements

When creating elements in the design, the way you create the elements determines how Authentic View will respond to user actions like pressing the Tab key and clicking the `Add...` prompt. The basic issue is what elements are created in Authentic View when an element is added by the user. For example, when the user adds an element (say, by clicking the element in the Elements Entry Helper), what child elements are created automatically?

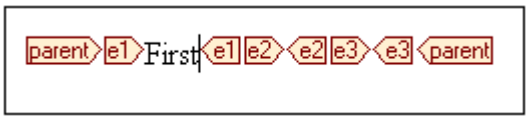
The most important point to bear in mind is that Authentic View follows the structure specified in the underlying schema. In order to ensure that Authentic View implements the schema structure correctly there are a few design rules you should keep in mind. These are explained below.

Unambiguous content model

A content model is considered unambiguous when it consists of a single sequence (with `maxOccurs=1`) of child elements (with no choices, groups, substitutions, etc). In such cases, when the element is added, the sequence of child elements is unambiguously known, and they are automatically added. In the screenshot example below, the three child elements are all mandatory and can occur only once.



When the element `parent` is added in Authentic View, its child elements are automatically inserted (*screenshot below*). Pressing the tab key takes you to the next element in the sequence.



If the `e2` element were optional, then, when the element `parent` is added in Authentic View, the elements `e1` and `e3` are automatically inserted, and the element `e2` appears in the Elements Entry Helper so that it can be inserted if desired (*screenshot below*). Pressing the tab key in `e1` takes the user to `e3`.



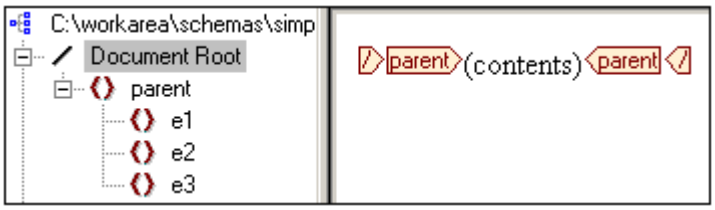
The above content model scenario is the only scenario Authentic View considers unambiguous. All other cases are considered ambiguous, and in order for Authentic View to disambiguate and efficiently display the desired elements the design must adhere to a few simple rules. These are explained below.

Ambiguous content model
For Authentic View to correctly and efficiently display elements correctly while an XML document is being edited, the StyleVision Power Stylesheet must adhere to the following rules.

- Child elements will be displayed in the order in which they are laid out in the design.
- In order for Authentic View to disambiguate among sibling child elements, all child elements should be laid out in the design document in the required order **and within a single parent node**. If the sibling relationship is to be maintained in Authentic View, it is incorrect usage to lay out each child element of a single parent inside multiple instances of the parent node.

These two rules are illustrated with the following example.

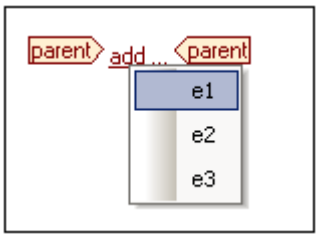
We consider a content model of an element `parent`, which consists of a single sequence of mandatory child elements. This content model is similar to the unambiguous content model discussed above, with one difference: the single sequence is optional, which makes the content model ambiguous. If you create a design document as shown in the screenshot below, there will be ambiguity in Authentic View.



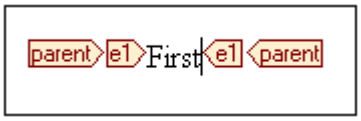
The Authentic View of the `parent` element will look like this (since the sequence is optional):



Clicking add... pops up a menu of the three child elements:

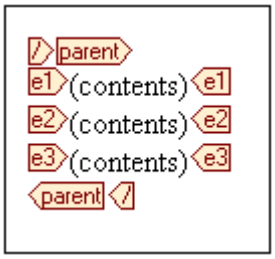


If you select one of these elements, it will be inserted (*screenshot below*), but since Authentic View cannot disambiguate the sequence it does not insert any of the remaining two elements, nor does it offer you the opportunity of inserting them:

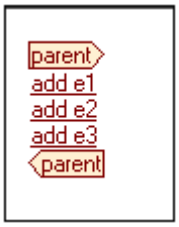


The **correct** way to design this content model (following the rules given above) would be to explicitly create the required nodes in the desired order within the single parent node. The design document would look like this:

 © 2005 Altova GmbH



Note that all three child elements are placed **inside a single parent node**. The design shown above would produce the following Authentic View:



The Authentic View user clicks the respective `add element` prompt to insert the element and its content.

Note:

- If an element can occur multiple times, and if the rules above are followed, then the element appears in the Entry Helper till the number of occurrences in Authentic View equals the maximum number of occurrences allowed by the schema (`maxOccurs`).
- Creating each child element inside a separate parent node (*see screenshot below*) not only creates isolated child–parent relationships for each child element so instantiated; it also increases processing time because the parent node has to be re-traversed in order to locate each child element.



4.6 Data-entry devices

Function
StyleVision enables you to add data-entry devices into a StyleVision Power Stylesheet. In Authentic View, when data is entered into a data-entry device, it is inserted into the XML document as element content or as an attribute's value. Data-entry devices enable you to constrain the data that goes into the XML document, and are, therefore, a useful way of improving data integrity.

General mechanism
Given below is a list of data-entry devices you can use, together with an explanation of how data is entered in the XML file for each device.

Data-Entry Device	Data in XML File
Input Field (Text Box)	Text entered by user
Multiline Input Field	Text entered by user
Combo box	User selection mapped to value
Check box	User selection mapped to value
Radio button	User selection mapped to value
Button	User selection mapped to value

The values entered in the text input fields are entered directly as XML content.

For the other data-entry devices, the Authentic View user's selection is mapped to a value. StyleVision enables you to define the list of options the user will see and the XML value to which each option is mapped. Typically, you will define the options and their corresponding values in a dialog.

Usage
To create a data-entry device, do the following:

1. Drag a node from the Schema window into the Design Window and drop it at the desired location.
2. From the context menu that appears, select the data-entry device you wish to create the node as.
3. For some data-entry devices, a dialog pops up. In these cases, enter the required information in the dialog, and click OK.

To **reopen and edit** the properties of a data-entry device, right-click **the data-entry device** (not the node containing it), and select the menu option **Edit Properties.** Alternatively, click **Properties | Selected object...**.

© 2005 Altova GmbH

Note:

- Data can be entered in data-entry devices only in Authentic View.
- Data-entry devices can also be created by changing the current component type of a node to a data-entry device. See Change to.
- In the HTML output, the visible entry that has been selected by the user is displayed in the output. Changing the value of a data-entry device in the HTML file does not change the text value in either the XML file or HTML file.

See also: Context menus.

4.6.1 Input fields

You can insert an Input Field or a Multiline Input Field in your StyleVision Power Stylesheet when you drop a node from the schema tree into the Design Window. The text that the Authentic View user enters into these fields is entered into the XML node for which the field was created.

Editing the dimensions of input fields
You can modify the dimensions of fields in two ways: by dragging the border/s of the field or entering a value in the Edit Properties dialog for the field (which you access by right-clicking the field (not the node) and selecting **Edit Properties...**). For Input Fields, the width can be modified (drag the right-border or, in the Edit Properties dialog, enter the number of characters the field should accommodate). For Multiline Input Fields, both width and height can be modified (drag the right or bottom border, or enter the required values in the Edit Properties dialog).

© 2005 Altova GmbH

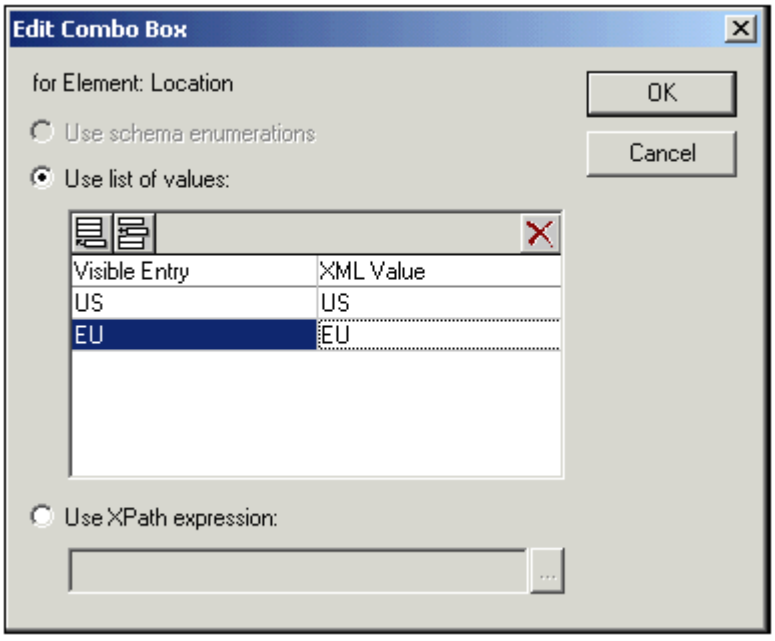
4.6.2 Combo boxes

A combo box presents the Authentic View user with a list of options in a drop-down list. The selected option is mapped to a value that is entered in the XML document. The mapping of drop-down list entry to XML value is specified in the StyleVision Power Stylesheet.

Mappings can be made in one of three ways:

- From the schema enumerations for the selected node. In this case, the visible entry (in the drop-down list) will be the same as the XML value.
- From a list defined in the Edit Combo Box dialog. You enter the visible entry and the corresponding XML value, which may be different.
- From a nodeset in the XML document, which is selected with an XPath relative to the current node. The nodes in the selected nodeset are displayed as the entries of the drop-down list, and the list entry selected by the Authentic View user is entered as the value of the node. The node that you create as the combo box is important. For example, say you have a `NameList` element that may contain an unlimited number of `Name` elements, which themselves have `First` and `Last` children elements. If you create the `Name` element as a combo box, and select the `Last` child element in the combo box, then, in Authentic View, you will get as many combo boxes as there are `Name` elements and each combo box will have the `Last` child as its dropdown menu entry. In order to get a single combo box with all the `Last` elements in the dropdown menu list, you must create the single `NameList` element as the combo box, and select the `Last` element in the XPath expression.

The Edit Combo Box dialog is shown below.



To define the entries and values for the combo box, do the following:

1. Select the method with which you wish to define the entries and values by clicking the appropriate radio button.
2. If you select Schema Enumerations, the enumerations are entered in automatically. If you select Use List of Values, you can insert, append, edit, and delete any number of drop-down list entries with their corresponding XML values. If you wish to use a nodeset from the XML file, select Use XPath Expression, and enter or build an XPath to select the nodeset.
3. Click OK to finish.

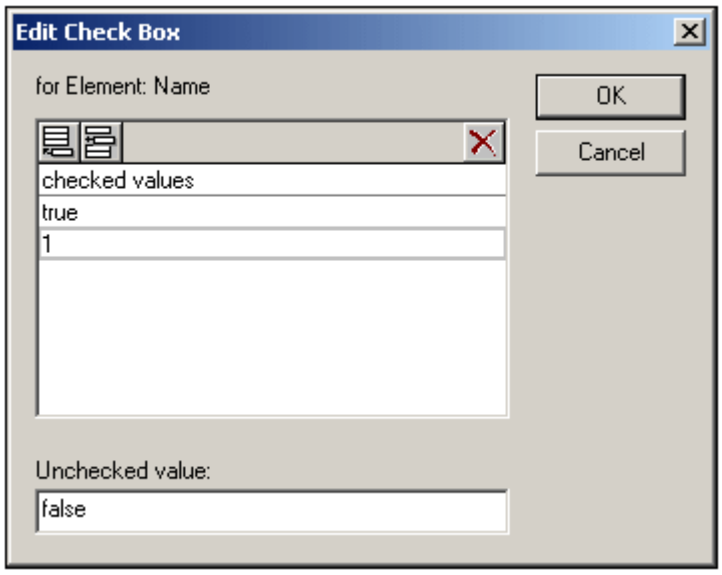
Note

- Using XPath to select the items of the combo box drop-down list enables you to create combo boxes with dynamic entries from the XML file itself.
- To **reopen and edit** the properties of a data-entry device, right-click **the data-entry device** (not the node containing it), and select the menu option **Edit Properties.** Alternatively, click **Properties | Selected object...**.
- To edit the properties of the combo box, in Design View, right-click the combo box (not the node containing the combo box), and click Edit Properties.

© 2005 Altova GmbH

4.6.3 Check boxes

You can create a check box as a data-entry device. This enables you to constrain user input to one of two choices. In the Edit Check Box dialog (shown below), you specify the XML values to map to the checked and unchecked events.



In the above screenshot, an element called `Name` has been created as a check box. If the Authentic View user checks the check box, a value of `true` will be entered as the value of the element `Name`. If the value is unchecked, then the value `false` is entered as the XML value of `Name` (as defined in the dialog).

To edit the properties of the check box, right-click the check box (not the node), and select **Edit Properties...** from the context menu that pops up.

Note: When a new `Name` (or check box) element is created in Authentic View, its XML value is empty (it is not the Unchecked Value). The Unchecked Value is entered only after the check box has first been checked, and then unchecked. To have a default value in a node, create a Template XML file that contains the default value.

4.6.4 Buttons

There are two types of button: radio buttons and buttons. Radio buttons allow the Authentic View user to enter data into the XML file. Buttons **do not allow** data-entry in Authentic View, but are useful for triggering events in the HTML output.

Radio buttons
Inserting radio buttons in the StyleVision Power Stylesheet allows you to give the user a choice among multiple alternatives. Each radio button you insert maps to one XML value. The way to use this feature is to create the node for which the data-entry is required multiple times as a radio button. For each radio button enter (i) some static text to indicate its value to the user, and (ii) a different XML value. The radio buttons for a node are mutually exclusive; only one may be selected at a time, and the associated XML value is entered as the value of the node.

Buttons
The button option allows you to insert a button and specify the text on the button. This is useful if you wish to associate scripts with button events in the generated HTML. Note, however, that a button does not map to any XML value and does not allow data entry in Authentic View.

© 2005 Altova GmbH

4.7 Lists

Function
If an element has multiple instances, to output its instances as the items of a list. A list can be bulletted or numbered. A list can be a static list or a dynamic list. Static lists (in which all list content comes from the StyleVision Power Stylesheet) are described first; the rest of the section is about dynamic lists (the content of which comes from the XML document).

Static lists
The content of a static list is contained entirely in the StyleVision Power Stylesheet. To create a static list, click **Insert | Bullets and Numbering**, or the Bullets icon or Numbering icon in the toolbar. This creates the first item of the list. Type in the content of the list item, and press enter to create a new list item. Static lists can be created for a single level. To change from bullets to numbering, click the Numbering icon; and vice versa.

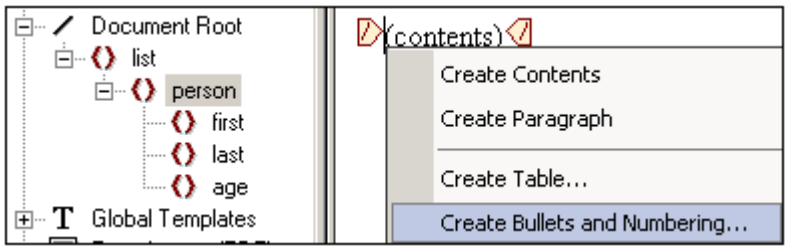
Dynamic lists
Dynamic lists display the content of a set of sibling nodes of the same name, with each node represented as a single list item in the list. The element, the instances of which are to appear as the list items of the list, is created as the list. The mechanism and usage are explained below.

General mechanism

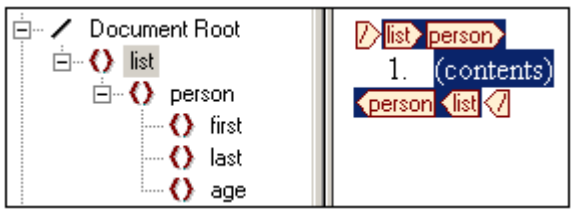
- Any element can be created as a list.
- When an element is created as a list, the instances of that element are created as the items of the list. For example, if in a `department` element, there are several `person` elements (i.e. instances), and you wanted to create a list of all the persons in the department, then you must create the `person` element as the list.
- Once the list has been created for the element, you can modify the appearance or content of the list or list item by inserting additional static or dynamic content such as text, Auto-Calculations, dynamic content, etc.

Usage: Basic

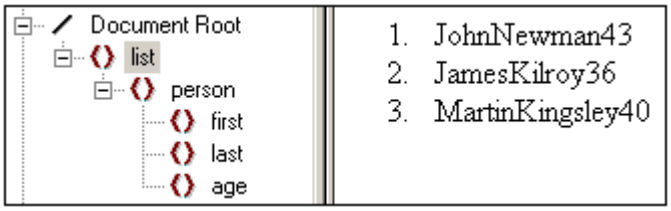
1. Create the required element as a list. In the example below, the element `person` is created as the list.



After you select the appropriate list formatting (bullets or numbering), this part of your Design Document will look like this:

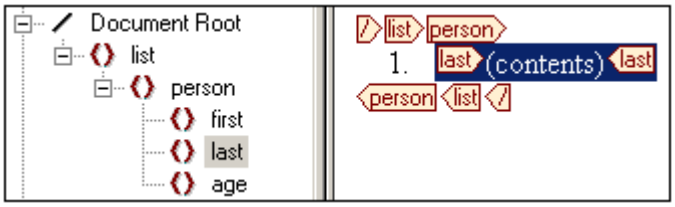


The Authentic and HTML Previews will look like this:

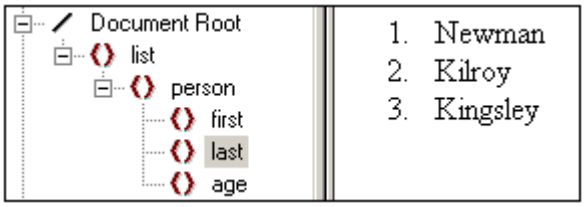


Notice that a list item has been created for each instance of the `person` element, and that the content of the **entire person element** (which includes content of all child elements) has been output.

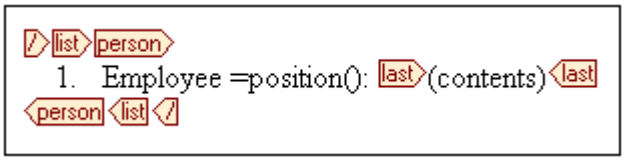
2. To modify the content of a list item, add new content or text after the list item marker. For instance, to list only the last name of persons, in the Design Document, drop the `last` element from the schema tree as contents into the list item. The contents of `//person/last` replaces the contents of the `person` element in the list item.



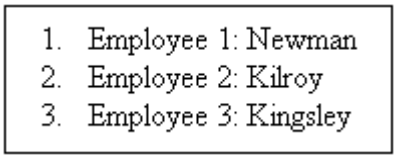
The Authentic and HTML Previews looks like this:



3. You could modify the list item further, as follows. Add the text shown in the illustration below. The equals to symbol indicates an Auto-Calculation; it returns the position of the current `person` element in the node-set.



The HTML Preview looks like this:

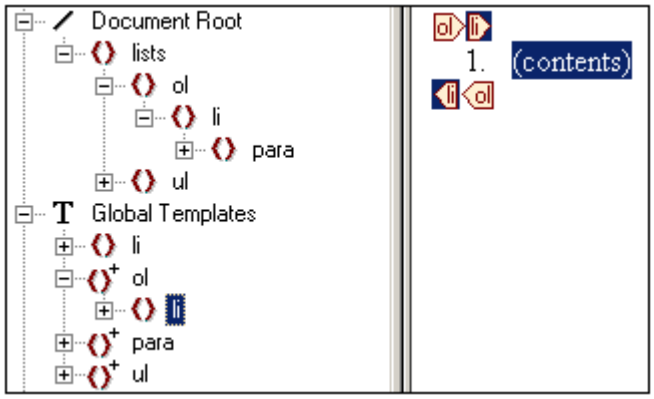


Usage: Advanced
The following issues, which are related to more complex list structures in schemas, are considered:

- Schemas often have separate elements for numbered lists and for bulletted lists. However, they usually have a common listitem-type element. An example is the HTML language, in which both the `ol` and `ul` elements have `li` as the listitem element. In this kind of scenario, it is best to define global templates for both the `ol` and `ul` elements,

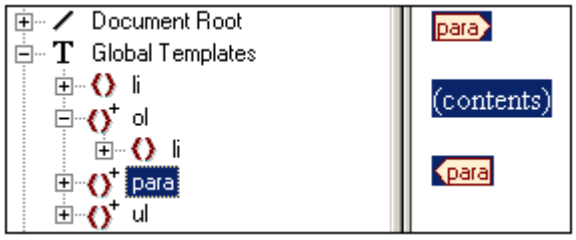
 © 2005 Altova GmbH

making sure to create the correct element as the list. Shown below is a view of the Design Document for the global template of an `ol`-type element.



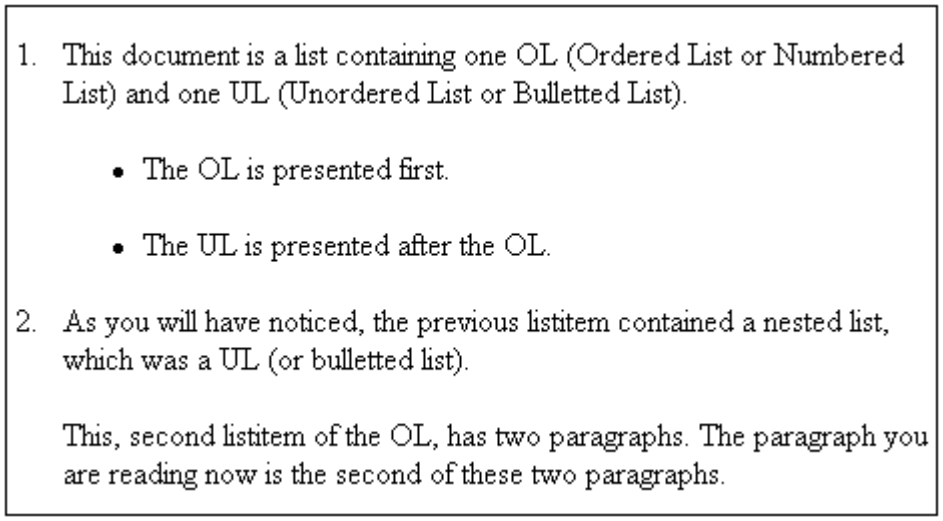
The `li` element was dragged from the content model of the `ol` element in the Global Templates list in the Schema Window, and was created as a numbered list. Similarly, create a global template for the `ul`-type element (but create the `li` element as a bulleted list in this case).

- The schema might have a `para`-type element as a child of the `li`-type element, and an `li`-type element may have more than one `para`-type element within it. The best way to deal with this kind of situation is to first define the `para`-type element as a global element in the XML Schema (this step is not necessary for DTDs). Then create a global template for the `para`-type element, making sure to create it as a paragraph. Shown below is a view of the Design Document for the global template of a `para`-type element.



If such a `para`-type element has been created as a block component, then pressing the Enter key in Authentic View causes another `para`-type element to be appended within the same listitem. Otherwise, pressing the Enter key causes another listitem to be appended.

- If nested lists are allowed by the schema, you must define global templates for the list-type elements (`ol`, `ul`, etc), making sure to create the listitem-type element/s as lists. Given below is the HTML Preview of part of a list that includes nested lists.

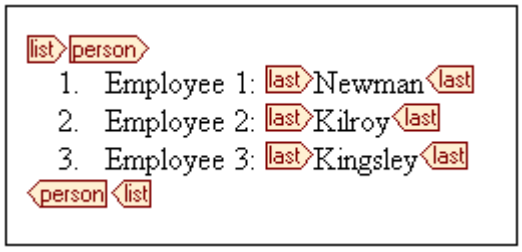

1. This document is a list containing one OL (Ordered List or Numbered List) and one UL (Unordered List or Bulletted List).
   - The OL is presented first.
   - The UL is presented after the OL.
2. As you will have noticed, the previous listitem contained a nested list, which was a UL (or bulletted list).

   This, second listitem of the OL, has two paragraphs. The paragraph you are reading now is the second of these two paragraphs.

- Note that in order to insert or append a listitem in Authentic View, the user must place the cursor in a listitem of the appropriate list. For example, if the cursor is in a listitem of a nested list, appending a listitem will append the listitem to this nested list. To append a listitem to the top-level list, the user must place the cursor in a listitem of the top-level list before appending a listitem.

Limitations

- In the Show Large Markup and Show Small Markup modes of Authentic View, the tags for listitem-type elements are not shown for the individual list items; instead only one pair of start and end tags is shown at the beginning and end of the entire list.



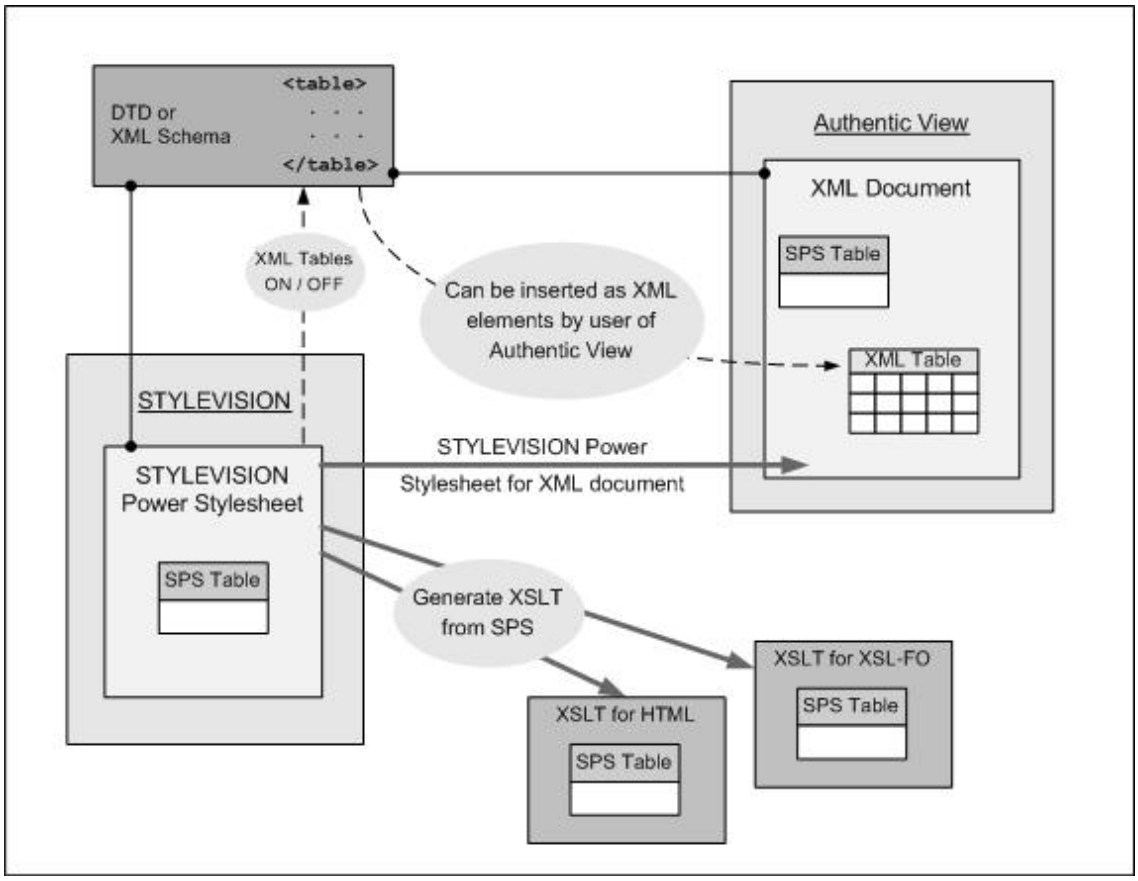
  In the figure above, only one pair of `person` tags occur—at the beginning and end of the list—when, in fact, there are three separate `person` elements (the listitems in this list).
- Only styles displayed in the Bullets and Numbering dialog are currently supported. User-defined bullets and numbering is currently not supported.
- For numbering nested lists so that list items have a form like `3.4.1`, there is no option for such a format you would like in the Bullets and Numbering dialog. To do this, you will have to use Auto-Calculations in a recursion mechanism. For details, see Numbering elements recursively.

 © 2005 Altova GmbH

4.8 Tables

In a StyleVision Power Stylesheet, two types of tables are used: **SPS tables** and **XML tables**. There are crucial differences between the two types, and it is important to understand these. A detailed description of SPS tables and XML tables, and instructions about how to use them, follows in the next sections. For now, we look at the broad picture.

The illustration below shows the relationship of SPS tables and XML tables to the StyleVision Power Stylesheet and to the XML document.



SPS tables
An **SPS table** is a component of a StyleVision Power Stylesheet, and is created and formatted using StyleVision. If present in a StyleVision Power Stylesheet, an SPS table appears in Authentic View as well as in the XSLT stylesheets you can generate with StyleVision.

An SPS table has no pre-determined structure. The structure is specified by the person who designs the StyleVision Power Stylesheet. An SPS table can be created anywhere in a StyleVision Power Stylesheet, and any number of SPS tables can be created.

The **structure** of an SPS table is **not represented as XML elements** in the XML document. SPS tables are entirely presentational devices. In whatever kind of document an SPS table is rendered, the table's structure is specified not in the markup vocabulary of the XML document but in the respective presentational vocabulary. SPS tables occur in three types of output:

- Rendered in Authentic View; a vocabulary specific to Authentic View is used to mark up SPS tables
- In StyleVision-generated XSLT stylesheets for HTML output, SPS tables are marked up

as HTML tables
- In StyleVision-generated XSLT stylesheets for XSL-FO output, SPS tables are marked up as XSL-FO tables

There are two types of SPS tables:

- **Static tables** are built up, step-by-step, by the person designing the StyleVision Power Stylesheet. After the table structure is created, the content of each cell is defined separately. The content of cells can come from random locations in the schema tree and even can be of different types. It is important to realize that the rows of a static table are not intended to represent a repeating data structure. This is why the table is said to be static: it has a fixed structure that does not change with the XML content.
- **Dynamic tables** are intended for data structures that repeat, and they are based on schema elements that have a substructure (at least one child attribute or element). Any element with a substructure repeats if there is more than one instance of it. Each instance of the element would be a row in the dynamic table, and all or some of its child elements or attributes would be the columns. In Authentic View, a dynamic table's structure reflects the content of the XML file and changes dynamically with the content.

XML tables
An XML table is created by the **Authentic View user** as a data structure in the XML document. The purpose of XML tables is to give the Authentic View user the option of inserting a table-type data structure in the XML document. This XML data structure can then be transformed to the table markup of the output format.

The data structure for an XML table must correspond to either the HTML or CALS table model. This XML table element and its content model must exist in the schema. An XML table can be inserted at any point in the XML document where it is allowed according to the schema. An XML table is formatted after it is inserted in the XML document.

Shown below is the Authentic View of an XML table that corresponds to the HTML table model.

Name	Phone
John Merrimack	6517890
Joe Concord	6402387

Data that is entered into the table's cells is entered as content of the corresponding XML elements. For example, the text view of the XML table shown in the illustration above looks like this:

```
<table border="1" width="40%">
  <tbody>
    <tr>
      <td>Name</td>
      <td>Phone</td>
    </tr>
    <tr>
      <td>John Merrimack</td>
      <td>6517890</td>
    </tr>
    <tr>
      <td>Joe Concord</td>
      <td>6402387</td>
    </tr>
  </tbody>
</table>
```

Note that the structure of the table exists as XML elements. An element in the XML table hierarchy can have any name; it is only important that the table structure correspond to the

 © 2005 Altova GmbH

HTML or CALS table model. The elements of the XML table (from the example above) could as easily be:

```
<phonelist border="1" width="40%">
  <items>
    <person>
      <name>Name</name>
      <phone>Phone</phone>
    </person>
    <person>
      <name>John Merrimack</name>
      <phone>6517890</phone>
    </person>
    <person>
      <name>Joe Concord</name>
      <phone>6402387</phone>
    </person>
  </items>
</phonelist>
```

Also note the following:

- An XML table can be inserted at any location in the XML document where, according to the schema, a table is allowed.
- In Authentic View, data is entered directly into table cells. This data is stored as the content of the corresponding XML table element.
- The formatting properties of an XML table are assigned in Authentic View.

Note that XSLT stylesheets generated with StyleVision will not contain XML tables—because the XML tables are not included in the StyleVision Power Stylesheet.

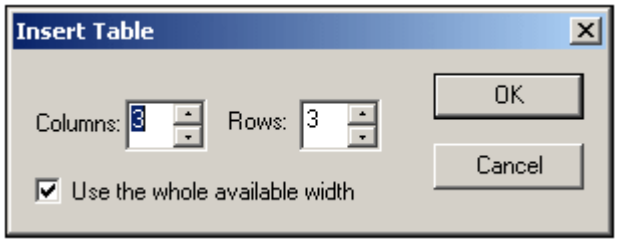
Summary for designer

From the document designer's perspective, the following points should be noted:

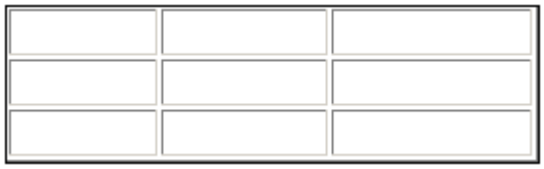
- An **SPS table** occurs in the XML document at a location determined by the designer of the document—not the user of Authentic View. The structure and formatting of SPS tables are specified by the designer of the StyleVision Power Stylesheet in StyleVision.
- The location, structure, and formatting of **XML tables** are specified by the user of Authentic View. The user may insert an XML table wherever this is allowed by the schema (remember: the table element corresponds to an element in the schema). Note also that the table format of XML tables is available only in Authentic View. The HTML output will not automatically display a table format. You will have to create your own template/s to match the table element, and manually add these to the generated XSLT stylesheets.

4.8.1 Creating static tables

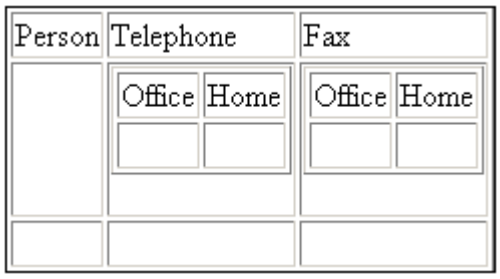
To insert a static table, click **Table | Insert table** or the  icon. The following dialog appears:



You can select the dimensions of the table and specify whether the table should occupy the whole available width. When you click OK, an empty table with the specified dimensions, as shown below, is created.



You can now enter content into table cells using regular StyleVision features. Cell content could be text, or elements dragged from the XML tree, or objects such as images and nested tables. The figure below shows a table containing nested tables.



Static SPS tables are especially well-suited for organizing XML data that is randomly situated in the schema hierarchy.

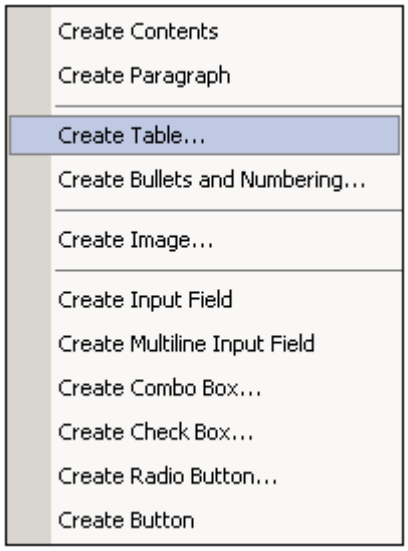
Deleting columns, rows, and tables
To delete a column, row, or table, place the cursor in the column, row, or table to be deleted, and click the menu item **Table | Delete Column**, **Table | Delete Row**, or **Table | Delete Table**, respectively. If you have nested tables, the table immediately containing the cursor will be deleted when the **Table | Delete Table** command is used.

Toolbar table editing icons
The table editing icons in the second row of the toolbar are shortcuts to the **Table** menu commands. These commands allow you to insert, delete, edit the structure of, and assign formatting properties to the static table. These icons can also be used for dynamic tables. They **cannot be used for XML tables**, since XML tables cannot be created in StyleVision but must be created in Authentic View by the user. XML tables can only be enabled in StyleVision.

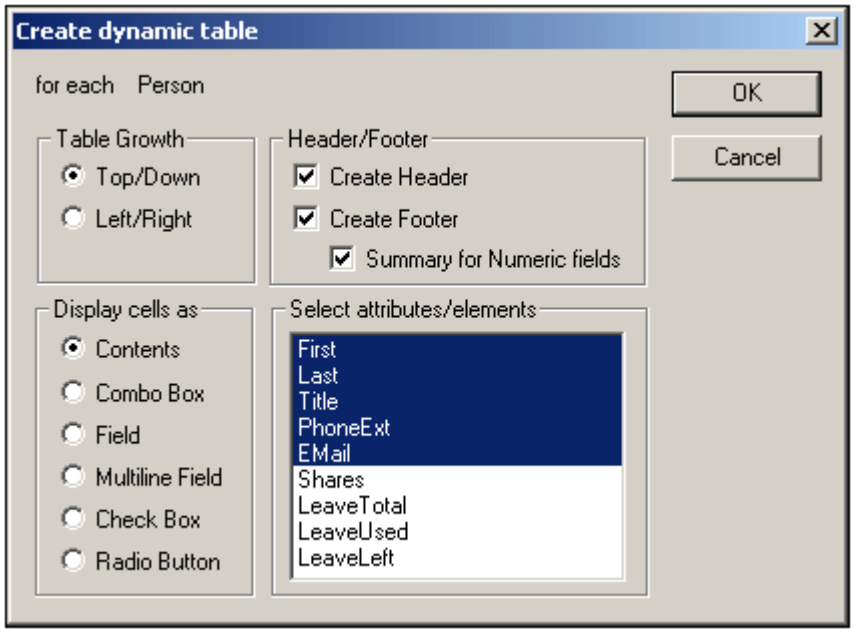
© 2005 Altova GmbH

4.8.2 Creating dynamic tables

Creating a dynamic SPS table
A dynamic SPS table is created by dragging an element from the XML tree into the Design window. On dropping the element into the desired location, the following context menu appears.



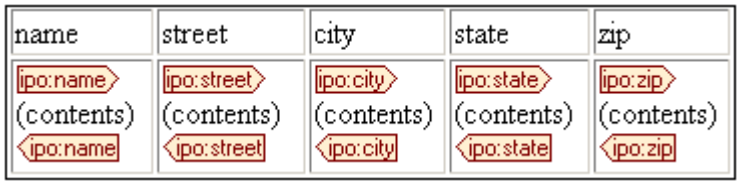
Click Create Table. This brings up the following dialog.



The child elements and attributes of the element that has been dragged into the Design window are displayed In the "Select attributes/element" list and can be created as columns of the table. Deselect the child nodes that you do not want and select any attribute/element you want to include as columns. (In the figure above, the elements `Shares`, `LeaveTotal`, `LeaveUsed` and `LeaveLeft` have been deselected.)

Note that columns are created only for child elements and attributes, but for no descendant on a lower level.

Table grows down or right
When a table grows top-down, this is what it would look like:



When a table grows left-right it looks like this:



Headers and footers
Columns can be given headers, which will be the names of the column elements. To include headers, check the Create Header check-box. To include footers, check the Create Footer check-box. Footers can only be created for tables that grow top-down. The footer of numeric columns will also sum each of these columns if the Summary for Numeric Fields check box is checked.

Header and footer cells can be joined and split, and rows can be inserted, appended, and deleted; this gives you considerable flexibility in structuring headers and footers. Additionally, headers and footers can contain any type of static or dynamic content, including conditional templates and auto-calculations.

Note: Headers and footers must be created when the dynamic table is defined. You do this by checking the Create Header and Create Footer options in the Create Dynamic Table dialog. Appending or inserting a row within a dynamic table does not create headers or footers but an extra row. The difference is significant. Real headers and footers are added to the top and bottom of a table, respectively. If a row is inserted or appended, then the row occurs for each occurrence of the element that has been created as a dynamic table.

Nested dynamic tables
You can nest one dynamic table within another dynamic table if the element for which the nested dynamic table is to be created is a child of the element that has been created as the containing dynamic table. Do the following:

1. Create the outer dynamic table so that the child element to be created as a dynamic table is created as a column.
2. In the dynamic table in Design View, right-click the child element.
3. Select **Change to | Table...**. This pops up the Create Dynamic Table dialog.
4. Define the properties of the nested dynamic table.

To nest a dynamic table in a static table, drag the element to be created as a dynamic table into the required cell of the static table. When you drop it, select **Create Table...** from the context menu that appears.

 © 2005 Altova GmbH

Tables for elements with text content
To create columns (or rows) for child elements, the element being created as a table must have a **child element or attribute node**. Having a **child text node** does not work. If you have this kind of situation, then create a child element called, say, `Text`, and put your text node in the `TableElement/Text` elements. Now you will be able to create `TableElement` as a dynamic table. This table will have one column for `Text` elements. Each row will therefore contain one cell containing the text node in `Text`, and the rows of the table will correspond to the occurrences of the `TableElement` element.

Contents of table body cells
When you create a dynamic table, you can create the node content as any one of a number of StyleVision components. In the examples above, the table body cells were created as contents. They could also be created as data-entry devices. There are two points to note here:

- The setting you select is a global setting for all the table body cells. If you wish to have an individual cell appear differently, edit the cell after you have created the table: right-click in the cell and, in the context menu that appears, select "Change to" and then the required cell content type.
- If you create cells as element contents, and if the element has descendant elements, then the content of the cell will be a concatenation of the text strings of the element and all its descendant elements.

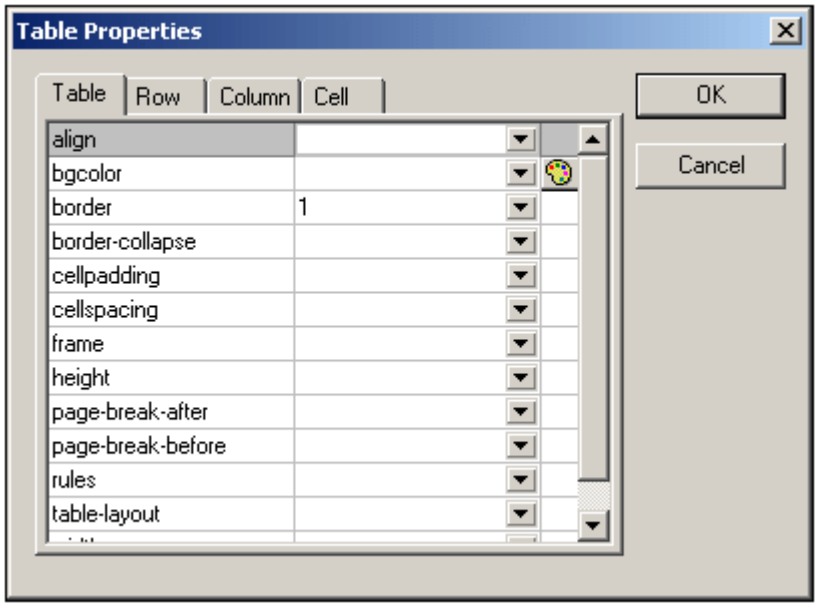
Deleting columns, rows, and tables
To delete a column, row, or table, place the cursor in the column, row, or table to be deleted, and click the menu item **Table | Delete Column**, **Table | Delete Row**, or **Table | Delete Table**, respectively. If you have nested tables, the table immediately containing the cursor will be deleted when the **Table | Delete Table** command is used.

Toolbar table editing icons
The table editing icons in the second row of the toolbar are shortcuts to the **Table** menu commands. These commands allow you to insert, delete, edit the structure of, and assign formatting properties to the dynamic table. These icons can also be used for static tables. They **cannot be used for XML tables**, since XML tables cannot be created in StyleVision but must be created in Authentic View by the user. XML tables can only be enabled in StyleVision.

Creating dynamic tables in global templates
You can also create dynamic tables in global templates. The process works in the same way as for the Root Template (given above). The important point to note is that, in a global template, a dynamic table can only be created for **descendant elements** of the global template node; it cannot be created for the global template node itself. For example, if you wish to create a dynamic table for the element `authors` within a global template, then this dynamic table must be created within the global template of the parent element of `authors`. It cannot be created within the global template of the `authors` element.

4.8.3 Formatting static and dynamic tables

After they have been created, static and dynamic tables can be formatted to a high degree of detail and precision. The formatting mechanism is the same for both types of table. Formatting properties are available at the table, row, column, and cell levels. To assign formatting properties, place the cursor anywhere inside the table, row, column, or cell you wish to format, and click either **Table | Table Properties...** or **Properties | Table...** or the  icon. The following dialog appears:



Click the required tab (Table, Row, Column, or Cell), enter a value for the formatting property you want to set, and click OK.

Cellpadding
The following points about cellpadding in tables should be noted:

- Cellpadding can be specified globally for all cells of the table in the Table Properties dialog.
- Cellpadding for individual cells can be specified by placing the cursor in the relevant cell and specifying padding (separately for each of the four components: left, top, right, and bottom) in the Layout tab of the Block Style window.

Additional cell formatting
In addition to the cell formatting available in the Table Properties dialog, additional properties are available in the Block Style Window. Place the cursor in the cell to be formatted and specify the value of the required property. For example, if you wish to put a right-border on a cell, place the cursor in the cell and, in the Layout tab of the Block Style Window, specify the color, style, and width for border-right.

Numbering the rows of a dynamic table
You can number the rows of a dynamic table by using the `position()` function of XPath. To do this, first insert a column in the table to hold the numbers, then insert an Auto-Calculation in the cell of this column with an XPath of: `position()`. Since the context node is the element that corresponds to the row of the dynamic table, the `position()` function returns the position of

 © 2005 Altova GmbH

each row element in the set of all row elements.

4.8.4 XML tables

An XML table is defined as a hierarchical XML structure, the elements of which contain the cell content of the table. This XML structure must correspond exactly to the CALS or HTML table model. In order for **users of Authentic View** to be able to insert XML tables the following two conditions must be fulfilled:

- An element must exist in the schema (DTD or XML Schema) with a content model corresponding either to the HTML or CALS table model
- XML tables must be enabled in the StyleVision Power Stylesheet

Note: The purpose of XML tables is to give the **user of Authentic View** the option of entering data as a table in Authentic View. This data **will be displayed as a table in Authentic View but will not automatically be displayed as a table in HTML output**. This is because no default processing for the XML table elements is defined. In order to obtain table formatting in the HTML output, you must either (i) define a global template for the table element, or (ii) manually define your own templates to provide processing for the XML table elements, and add these templates to the generated XSLT files. If you wish to have a table in your HTML output, we recommend that you use static and/or dynamic SPS tables.

Defining the table content model in the schema
The `table` element in the schema must have a content model corresponding to either the HTML or the Exchange model subset of the CALS table model. The content model of the `table` element in your schema **must correspond exactly** with either of these two table models, i.e. all elements and attributes defined in the table model must be correspondingly present in the element content model. For information about the CALS table model, see the CALS table model at OASIS. For an example of a `table` element having an HTML table structure, see the HTML-OrgChart XML Schema in the Examples folder (`HTML-OrgChart.xsd`).

A table model corresponding to the HTML table model would have a structure as shown in the XML fragment below.

```
<table>
  <tbody>
    <tr>
      <td/>
    </tr>
  </tbody>
</table>
```

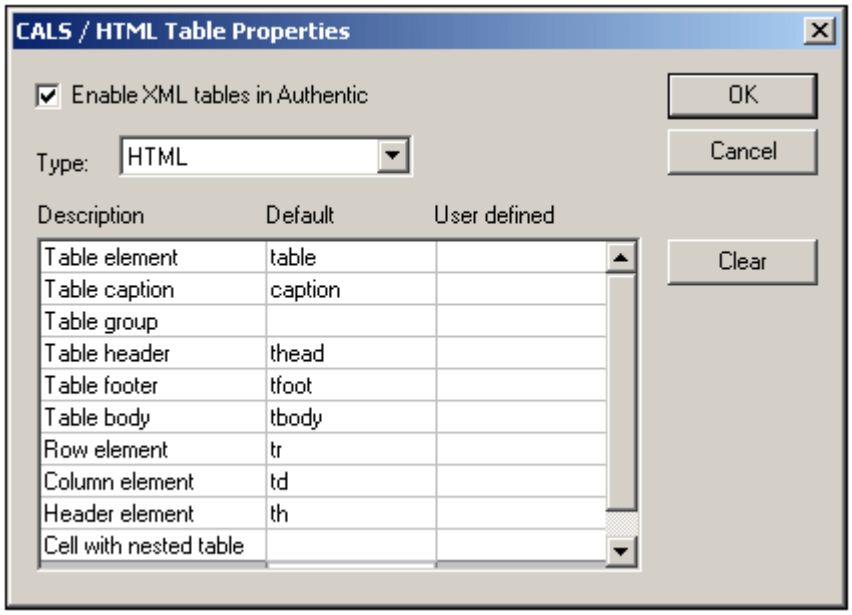
The element names in the example above are the default names you will find in the StyleVision Power Stylesheet. If the names of the table elements in your schema do not match these default names, you must map the names of the table elements in your schema to the default names in the Table Properties dialog of StyleVision. You can do this when you check the Enable XML tables option in the dialog. If there is more than one element in the schema that has a valid table content model, then the mapping to the default names determines which element will be used as the table element.

Caution: If an element called `table` exists in the schema, it will be treated as the XML table element if XML tables have been enabled in the StyleVision Power Stylesheet (because `table` is the default name for the table element in the StyleVision Power Stylesheet). This could lead to errors if the element `table` is not intended to be used as a table element.

Enabling XML tables with StyleVision
In order for the user of Authentic View to be able to create XML tables in an XML document, XML tables must be enabled in the StyleVision Power Stylesheet.

To enable XML tables, click **Authentic | CALS/HTML Tables...**. This pops up the CALS / HTML Table Properties dialog shown below.

 © 2005 Altova GmbH

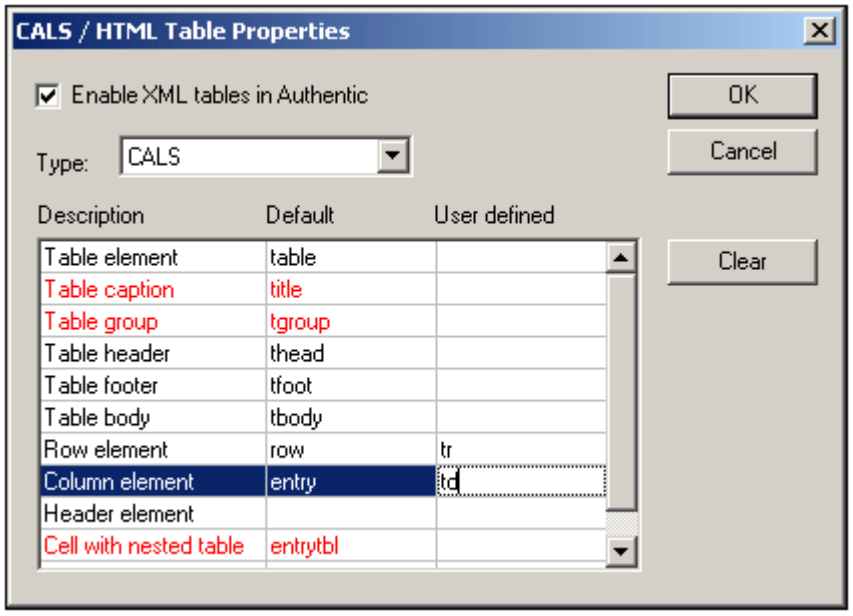


Now check the "Enable XML tables in Authentic" check-box.

Then select the table model type: either HTML or CALS. The screenshot above shows the default names for the HTML table model. If a default element name matches the corresponding table element name in your schema, then the default name is colored black, otherwise red. The screenshot above indicates the following about the schema:

- there is an element called `table` with a content model the same as the HTML table model (all default names appear black)
- all the default HTML elements exist in the content model of `table` (not one of the default names appears red)

The content model of a `table` element that follows the HTML model would not correspond with the CALS table model, which is different. Given below is a screenshot of the CALS / HTML Table Properties dialog (for the same schema as above) with the table model type set to CALS.



Notice the following:

- Four elements in the HTML content model are not present in the CALS table model (`caption`, `th`, `tr`, `td`). These elements do not appear as default elements in the dialog when the XML table type is switched to CALS.
- The default CALS table elements that do not exist in the schema are colored red. Those CALS table model elements that do exist in the `table` element's content model appear black.
- In the CALS table model, the default names of the row element and cell (or column) element are `row` and `entry`, respectively. If no mapping to schema elements is defined, then these element names are displayed in red. In the dialog, however, `row` and `entry` have been mapped to `tr` and `td`; this causes both the Default and User defined elements to be displayed in black.

If you wish to use the CALS table model, you should alter the schema to properly include the missing elements and attributes. Once you have selected the appropriate table model, click OK. XML tables are now ready to be used in Authentic View.

Note:
The following general points about XML tables should be noted:

- The `table` element can only be inserted at locations in the XML document where the schema allows the table element.
- You, as the person who designs the StyleVision Power Stylesheet, only enables XML tables. It is the Authentic View user who inserts an XML table at his or her discretion.
- An XML table is structured and formatted in Authentic View, that is, by the Authentic View user. The formatting of an XML table cannot be controlled through the StyleVision Power Stylesheet.
- In the CALS table model, the `cols` attribute of the Table group element (`tgroup`) specifies the number of columns in the table. This attribute–value pair is entered (both attribute and value) into the XML when the Authentic View user inserts an XML table. This is because the user must specify the number of columns when inserting the table is inserted. The value of this attribute changes automatically whenever a column is inserted, appended, or deleted using the GUI tools. The `cols` attribute is therefore not shown in the Attributes Entry Helper and its value, therefore, cannot be modified there.

© 2005 Altova GmbH

- Data entered in the cells of an XML table is entered as content of the `entry` element (in the case of a CALS table) or `td` element (HTML table), or corresponding user-defined element, as the case may be.
- To obtain the content of XML tables in HTML output, you must generate the XSLT (using a **File** menu command) and, in it, create a global template for the table element.

4.9 Auto-Calculations

Function
The Auto-Calculation feature (i) displays the result of an XPath evaluation at any desired location in the output document, and (ii) optionally updates a node in the XML document with the result of the XPath evaluation.

The Auto-Calculation feature is a useful mechanism for including:

- Calculations based on string or number operations on dynamic data
- Calculations based on the dynamic structure of the document (relative position of a node, etc)
- Inserting dynamic XML content at any location in the output document

Calculations based on string or number operations
The result of an XPath operation on dynamic node content can be displayed at any location in the document. The result can also be saved to an XML node if, in the Auto-Calculation dialog, you specify an XML node to update.

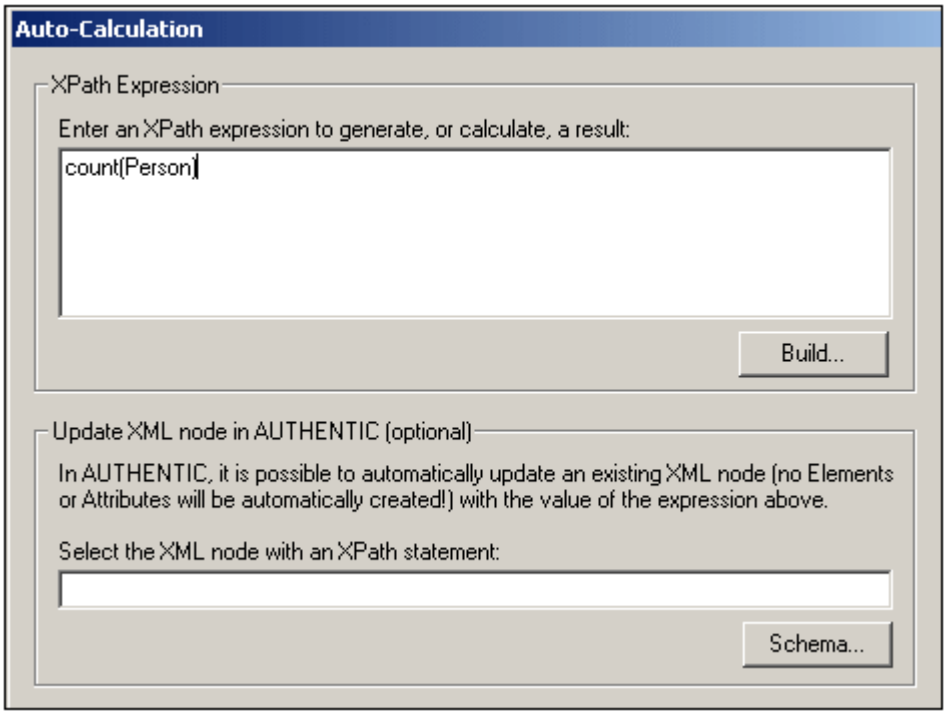
Calculations based on the dynamic structure of the document
Since an XPath expression can also operate on aspects of the document structure, the Auto-Calculation function can be used to display meta information about a dynamically changing document structure. For example, you can use it to count the number of `Person` elements in an `Office` element, and in this way obtain the total number of employees in an office. The result is dynamically updated as the number of `Person` elements changes. Or you can use the `position()` function of XPath to dynamically insert row numbers in a dynamic table.

Inserting dynamic XML content
You can use the Auto-Calculation function to insert content from any node in the document at another location, and to save this data to another node. The Auto-Calculation function is a useful way to display dynamic content of some node at another location. Note, however, that if a nodeset is selected by the XPath, only the content of the first node of the nodeset is displayed.

Usage
To insert an Auto-Calculation, do the following:

1. Place the cursor as an **insertion point** at the location where the Auto-Calculation result is to be displayed and click **Insert | Auto-Calculation**. In the submenu that appears, select Value if the result is to appear as plain text, Input Field if it is to appear within an input field (i.e. a text box), or Multiline Input Field if in a multiline text box.

 © 2005 Altova GmbH


Auto-Calculation
XPath Expression
Enter an XPath expression to generate, or calculate, a result:
count(Person)
Build...
Update XML node in AUTHENTIC (optional)
In AUTHENTIC, it is possible to automatically update an existing XML node (no Elements or Attributes will be automatically created!) with the value of the expression above.
Select the XML node with an XPath statement:
Schema...

2. Enter the XPath expression for the Auto-Calculation. **The XPath expression must be a dynamic expression,** i.e. it must include at least one node value as an operand. The result of the Auto-Calculation will be displayed at this location in the result document.
3. Optionally, select a node in the schema document to update.

Moving Auto-Calculations
You can move an Auto-Calculation to another location by clicking the Auto-Calculation (to select it) and dragging it to the new location. You can also use cut/copy-and-paste to move/copy an Auto-Calculation. Note, however, that the XPath will need to be changed if the context node in the new location is not the same as the previous one.

Updating an XML node
You can pass the result of an Auto-Calculation to a node in the XML file. To do this, in the Auto-Calculation dialog, enter an XPath to select the node to update. **For the node to be updated, two conditions need to be fulfilled:**

- The XPath expression for the Auto-Calculation must include the value of an XML node as one of the operands, i.e. a dynamic value. If the XPath expression uses a static value (such as a string), then the Auto-Calculation is invalid and no node will be updated.
- In Authentic View, one of the XML node values used as an operand in the Auto-Calculation expression must be modified. If none of the dynamic (XML) operands is modified, the Auto-Calculation result will not update the specified node in the XML file.

Updating an XML node with an Auto-Calculation that is inside a Condition
If you use the Update XML Node feature of **an Auto-Calculation that is inside a condition**, it is important to note the sequence of operations that occurs when a node value is changed:

1. A value is changed.
2. All **currently visible** Auto-Calculations (i.e. those Auto-Calculations in currently active conditions) are carried out first and node values are updated (if this is specified). Auto-Calculations which are within conditions that are not currently active **are not carried out**.

3. **After** Auto-Calculations are done, a condition may be activated that contains an Auto-Calculation. When this happens the result of the Auto-Calculation within this new condition is generated **on the fly** and displayed in Authentic View. But **no node is updated**—even if it is specified that a node be updated.

We step through an example to give you a better idea of how this works. Consider the following situation:

- Node A contains a conditional template with two conditions.
- Condition One says: if `B = 0`, display nothing
- Condition Two says: if `B = 1`, use an Auto-Calculation to display the value of `B` and to update the value of `A` with the value of `B`

The following list describes what happens when the value of `B` is changed. We start with `B = 0` and `A = 0`.

- When `B = 0`, no Auto-Calculation is visible (Condition One). So `B = 0` and `A = 0`.
- Now change the value of `B` to `1`. This corresponds to Step 1 above.
- The current condition is still Condition One (it will be changed only after Auto-Calculations are done). Since no Auto-Calculation is visible (Condition One), no Auto-Calculation is done. This corresponds to Step 2 above.
- Now, since `B = 1`, the condition is switched from Condition One to Condition Two. The new condition (Condition Two) does contain an Auto-Calculation, so the Auto-Calculation result is generated on the fly (the value of `B`), and displayed, but `A` is not updated. So `B = 1` and `A = 0`.
- Now go through the cycle again. Change the value of `B` back to `0`. (Step 1).
- The value of `B` is `0`. The current condition is Condition Two, which has an Auto-Calculation. So the Auto-Calculation is done. The result is the value of `B`, which is `0`. Node A is updated with this value. (Step 2).
- Next, the condition is switched: to Condition One, which specifies that nothing is to be displayed. So `B = 0` and `A = 0`. (Step 3).

The important point to grasp is that **Auto-Calculations with node updates** are done only once, at the beginning of the event sequence, and **it is only after Auto-Calculations are evaluated** that conditions are switched.

 © 2005 Altova GmbH

4.10 Conditional templates

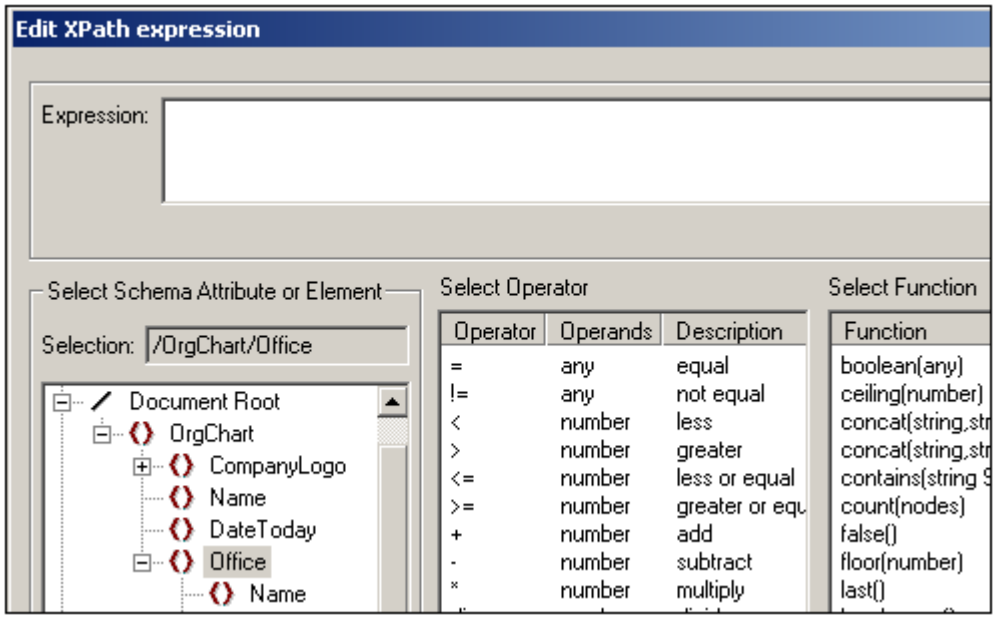
Function
To specify alternative processing according to the value of the XML content.

General mechanism

- A conditional template can be inserted anywhere in the Design Document, at a cursor insertion point or around a component—dynamic or static.
- The node **within which** the conditional template is inserted is the **context node**.
- A conditional template consists of one or more conditions.
- The condition to be tested is specified in an XPath expression.
- The **content** of the **first condition to be evaluated as true** is processed; the other conditions in the template are ignored—even if one or more of them evaluate to true. Note, therefore, that the order in which the conditions are listed is significant.
- Optionally, an **Otherwise condition** can be specified for the event that none of the previous conditions evaluates to true.
- If particular processing is required to match different conditions, i.e. if the conditions are not mutually exclusive, then separate conditional templates must be constructed for these conditions. The non-exclusive conditions should not be included in the same conditional template.

Usage

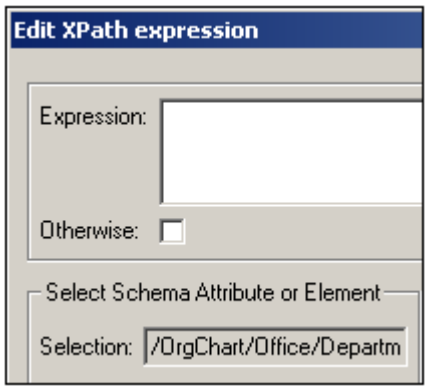
1. Place the cursor where the conditional output is required, or select the node or static content (or both) around which the condition is to be created.
2. Right-click, and select **Insert | Condition...**. The Edit XPath expression dialog appears. It looks like this:



   The context node is highlighted in gray.
3. Build the XPath expression by double-clicking schema nodes and XPath operators and functions. Alternatively, you can enter the XPath expression directly in the Expression field.
4. Click OK. The start and end of the conditional template are indicated by cyan tags with a question mark: ?Location = "... ?.
5. Place the cursor within the conditional template tags, and add the content (static or

dynamic) to be processed and output if the condition (specified in the XPath expression) is true. If a node or static content was selected in Step 1, the conditional template tags are placed around this document fragment, and that document fragment is the content of the conditional template.

6. To add an alternative condition, right-click either the start or end tag of the conditional template, and select **Condition | Add New** or **Condition | Add Copy**. In either case, the Edit XPath expression dialog appears, in which you enter the XPath expression to test for the alternative condition.
7. Click OK. Another condition is added to the conditional template. If you selected **Condition | Add New**, the new condition is empty, and you have to add the content to process and display when this condition is evaluated as true. If you selected **Condition | Add Copy**, then the content of the newly created condition is a copy of the content of the selected condition; you can modify this content as required.
8. To create, an Otherwise template, follow Step 6 to create another conditional template. In the Edit XPath expression dialog, check the Otherwise check box (shown below). (The Otherwise check box appears if one conditional template already exists on that node, static content, or document fragment.)



The Expression field is grayed out.

9. Click OK to create the Otherwise template, and then add or modify content for the template.

To change the order of conditions in a list of conditions in a conditional template, right-click the conditional template and use the **Condition | Move Up** and **Condition | Move Down** commands to set up the required order. The order of the conditions in a conditional template is significant. This is because the first condition in the list to be evaluated as true is processed; the other conditions are then ignored.

To move a conditional template to a new location, drag the conditional template to the desired location. If the context node of the new location is not the same as that of the original location, then correct the XPath in the condition/s of the conditional template as well as the location paths of any dynamic content within the condition template.

To delete a conditional template without removing the contents, drag the contents to a location outside the tags of the condition and then delete the now-empty condition.

Examples
The section Using Conditional Templates in the StyleVision tutorial has examples of conditional templates for:

- Setting up alternative data structures based on the value of a node
- Selecting and processing a node according to the value of another node
- Formatting text differently based on the numeric value of a node.

 © 2005 Altova GmbH

The section Creating the Table Footer in the StyleVision tutorial has examples of conditional templates for:

- Compiling a list based on the numeric value of a descendant node
- Inserting characters based on the position of a node in a nodeset.

Expanding and collapsing elements in Authentic View
You can use conditional templates to provide expanded and collapsed views of elements. To do this, carry out the following steps:

1. In the schema, define an attribute for the elements you want expanded/collapsed. For example, the attribute `collapse` with enumerations `yes` and `no`.
2. In StyleVision, create a conditional template around the elements to be expanded/collapsed. If the `collapse` attribute has a value of `yes`, then set the conditional template to display a collapsed form of the element. Otherwise display the descendants of the element.

Conditional Templates with Conditions that use Auto-Calculations
For a description of the issues involved, see Auto-Calculations.

4.11 Date Picker

Function
The Date Picker is a graphical calendar in Authentic View for entering dates in the ISO format (**YYYY-MM-DD**) in the XML document. The Date Picker can be enabled for any element of datatype `xs:date`. Including the Date Picker is useful since it provides a quick and easy way for the Authentic View user to enter a date in the fixed XML format.

General mechanism

- The Date Picker must first be enabled for one or more XML schema elements of datatype of `xs:date` in the StyleVision Power Stylesheet.
- In Authentic View, dates are entered via the Date Picker by clicking the required date.
- Dates entered via the Date Picker are stored in the standard XML Schema format **YYYY-MM-DD**.

Creating the Date Picker
In order to insert the Date Picker in your StyleVision Power Stylesheet, you must toggle on the Auto-Addition of the Date Picker. You can do this by toggling the  icon ON or toggling on **Authentic | Auto-add Date Picker**. When the Auto-Addition of the Date Picker is switched on, the Date Picker is inserted when any element of datatype `xs:date` is created as either contents or an Input Field, or changed to either of these two components.

To create an `xs:date` datatype element with an included Date Picker, do the following:

1. Ensure that the Auto-add Date Picker feature has been switched on.
2. Drag the `date` datatype element (called `Date` in the example below) from the schema tree and create it at the desired location in the Design Document as contents or an Input Field. The Design Document for such an element created as contents will look like this:

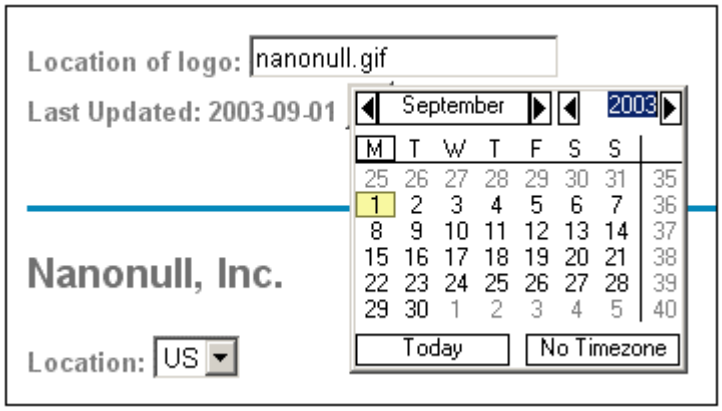
   The Date Picker is created.

Using this StyleVision Power Stylesheet, the Authentic View will look like this:

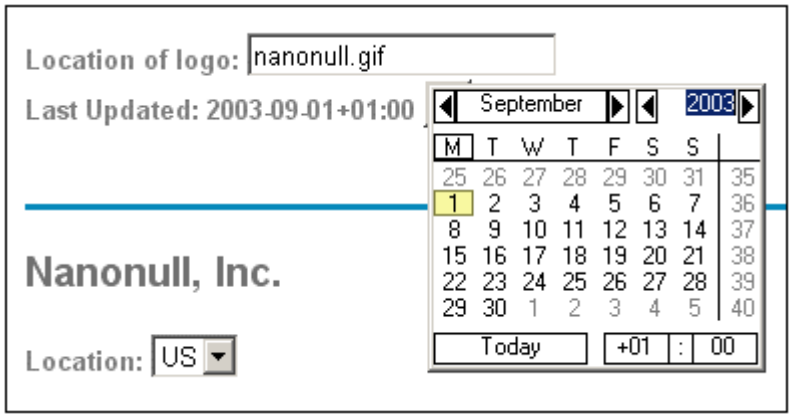


In Authentic View, the content of the Date element is displayed first (in the example above: `2003-09-01`), and after it the icon for the Date Picker. To modify the date, click the icon; this pops up the Date Picker (shown below):

© 2005 Altova GmbH



The Authentic View user can also enter the timezone by clicking the Timezone button, which is set to a default of No Timezone. The XML entries for the timezone are also in the standard XML Schema format for timezones, e.g. `2003-09-01+01:00`.



Note: If you have created the `xs:date` datatype element as contents or Input Field, but without the Date Picker, you can subsequently insert the Date Picker in your StyleVision Power Stylesheet by placing the cursor at the desired location inside the element, and clicking **Insert | Date Picker**. See Date Picker in the User Reference.

Limitation: The Auto-Calculation feature cannot be used for date-based operations. So you cannot, for example, use Auto-Calculation to calculate the number of days between two values of the `xsd:date` datatype.

4.12 Input Formatting

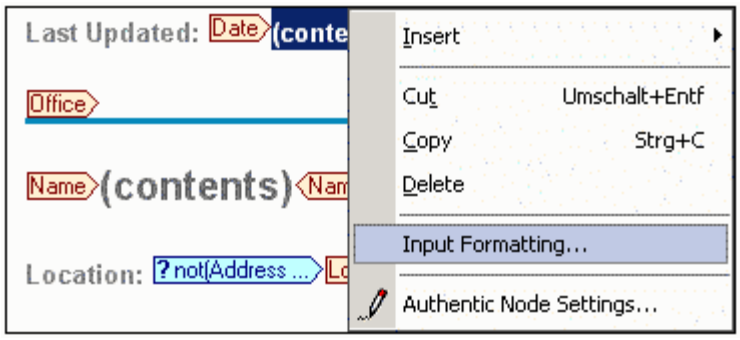
Function
With Input Formatting, the contents of numeric XML Schema datatype elements can be formatted for Authentic View and/or HTML output as required. Input Formatting can also be used to format the result of an Auto-Calculation.

General mechanism

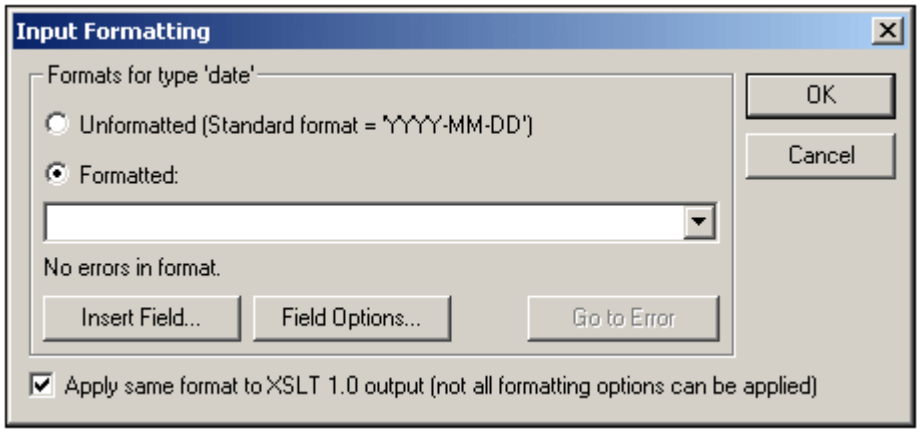
- The numeric XML Schema datatype element to be formatted must be present in the Design Document as contents or as an input field.
- Select the `(contents)` placeholder or input field, and apply the required formatting via the Input Formatting dialog. Formatting can also be applied to an Auto-Calculation field.
- The formatting applies to Authentic View, and optionally to HTML output. (The actual data in the XML file is stored in the standard format for that datatype.)

Applying input formatting
Input formatting can be applied to any numeric XML Schema datatype element, such as `xs:date`, which is present in the Design Document **as contents or an input field**. To format such an element, do the following:

1. In the Design Document, right-click the `(contents)` placeholder or input field of the element, and in the context menu that appears (shown below), click **Input Formatting...**.

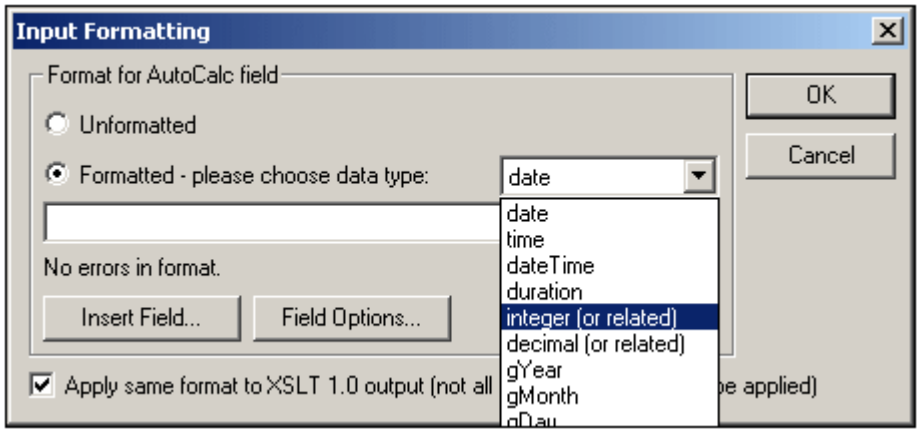


2. The Input Formatting dialog (shown below) appears. Note that the dialog below contains the line: "Formats for type 'date'" and that the standard format for the date datatype is given. This is because the datatype of the element selected in this case is `xs:date`. For another datatype, the information and options in the dialog would be correspondingly different.



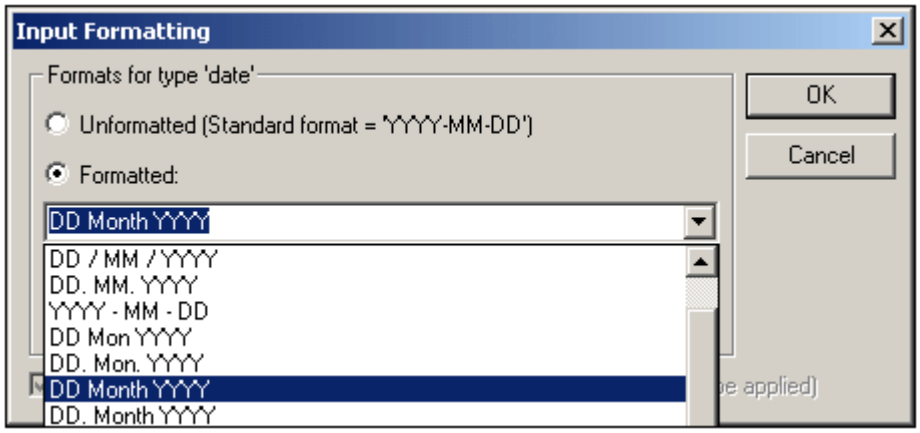
 © 2005 Altova GmbH

You can specify that the element's content be unformatted (i.e. that it have the standard formatting for that datatype), or that it have a customized formatting.
If you are formatting an Auto-Calculation field, the following dialog appears:



Since an Auto-Calculation field has no datatype assigned to it, you must select one from the drop-down menu shown in the figure above.

3. To specify a customized format, select the Formatted radio button, then enter a format in the input field. You can enter a format by selecting a pre-defined format from the drop down menu (shown below), by building one using the Insert Field and Field Options buttons, or by typing in the format. What pre-defined formats are available depends upon the datatype of the element or, in the case of an Auto-Calculation, the datatype you select.



For details of syntax, see the section below.

4. The formatting that you enter will apply to the Authentic View of an XML document. By default, the check box "Apply same format to XSLT 1.0 output" is checked, which means that the formatting will also be applied to HTML output via the generated XSLT stylesheets. If you wish to not have the formatting apply to HTML output, uncheck this check box. Note that some output formats cannot be applied to the HTML output. In these cases, the "Apply same format to XSLT 1.0..." option will be grayed out.
5. Click OK to apply the formatting.

Note:

- An error in syntax causes the OK button in the Input Formatting dialog to be disabled and the Go to Error button to be enabled. An error message is also displayed in the line below the combo box. Clicking the Go to Error button takes you the point in the format statement where the syntax error is.
- If the result of an Auto-Calculation does not match the fixed XML Schema format of that datatype, then the formatting will be undefined and the Auto-Calculation result will not be displayed correctly.

Insertion and Overwrite fields

Depending on the datatype, the Authentic View user will be able to insert or overwrite digits in the input field. Fields which do not have any semantic meaning beyond their numerical significance, such as integers and decimals, are insertion fields. Fields that do have some kind of semantic meaning, such as years and days, are overwrite fields.

Syntax for input formatting

The basic syntax is:

`[prefix character/s]field[suffix character/s][{field-option1,field-option2,...}]`

where **`prefix character/s`** and **`suffix character/s`** are optional specifiers used to control alignment and the display of positive/negative symbols; **`field`** can be any datatype-specific formatting or text; and **`{field-option(s)}`** is an optional qualifier, that enables additional formatting options.

- Any number of fields can be specified, and any two fields can be separated by the following characters: space, hyphen, comma, colon, period, or by a text string in single or double quotes. Note, however, that fields do not need to be separated.
- One or more optional field-options are allowed per field. The field-option/s for each field must be contained in curly braces, and must follow the field without any intervening space. Multiple options are separated by "`,`" (comma).

Example for an `xs:date` datatype element:

```
DD Month{uc,ro} YYYY
```

where `uc` and `ro` are field-options for making the `Month` field uppercase and read-only, respectively

An example of the output would be:

```
24 SEPTEMBER 2003
```

Example for an `xs:decimal` datatype element:

```
"$"(##0.00)
```

Examples of the output would be:

```
$ 25.00
$ 25.42
$267.56
```

 © 2005 Altova GmbH

Field types: You will find examples for the datatype-specific field types in the drop-down menu of the Input Formatting dialog. The following tables list the available field types.

Field Type	Explanation
#	Space if no digit at position
0	Zero if no digit at position
,	Digit separator
Y	Year
y	year (base = 1930); see Note below
M	Month
D	Day
W	Week number
d	Weekday number (1 to 7)
i	Day in the year (1 to 366)
h	Hour (0 to 23)
H	Hour (1 to 12)
m	Minute
s	Second
AM	AM or PM
am	am or pm
AD	AD or BC
ad	ad or bc
CE	CE or BCE
ce	ce or bce

Field Type	Explanation
Weekday	Weekday (Sunday, Monday...)
WEEKDAY	Weekday (SUNDAY, MONDAY...)
weekday	Weekday (sunday, monday...)
Wkd	Weekday (Sun, Mon...)
WKD	Weekday (SUN, MON...)
wkd	Weekday (sun, mon...)
Month	Month (January, February...)
MONTH	Month (JANUARY, FEBRUARY...)
month	Month (january, february...)
Mon	Month (Jan, Feb...)
MON	Month (JAN, FEB...)
mon	Month (jan, feb...)

Field length: The following points should be noted:

- When fields such as Y, M, and m are used, the exact length is output. When a field type is repeated, then the length is what is specified in the formatting. For example, if you specify YYYYYY, then the first two (redundant) spaces are filled with 0s. See also Base Year below.
- The Authentic View user is not allowed to add more digits than are specified in the format. If the length of a **pre-existing** value in the XML file exceeds the length specified in the format, the entire value is displayed; the user will, however, not be allowed to enter more digits.
- The * (asterisk) symbol is used to extend the length of a non-semantic numeric field (integers, decimals, etc). In the case of decimals, it can be used on either or both sides of the decimal point. For example, the input formatting *0.00* ensures that a number will have zeroes as specified in the formatting if these digit locations are empty, as well as any number of digits on both sides of the decimal point.

© 2005 Altova GmbH

Prefix and suffix characters: Prefix and suffix characters are used to specify alignment and details of the positive/negative symbol. The following table lists the prefix and suffix characters.

Prefix	Suffix	Explanation
<		Left aligned; default for text
>		Right aligned; default for numbers
?		Minus symbol if negative; nothing otherwise; default
?>		Minus symbol flush left if negative; nothing otherwise
<?>		Minus symbol flush left if negative; nothing otherwise
-	-	Minus symbol if negative; space otherwise; located before (prefix), after (suffix)
->	>-	Minus symbol if negative; space otherwise; located flush left (prefix); fixed position (suffix)
<->		Minus symbol flush left if negative; space otherwise
+	+	Plus or minus sign always; located before (prefix), after (suffix)
+>	>+	Plus or minus sign always; flush left (prefix), fixed position (suffix)
<+>		Plus or minus sign always; located flush left
(	)	Parentheses if negative; space otherwise
(>	>)	Parentheses if negative; space otherwise; located flush left (prefix); fixed position (suffix)
<(>		Parentheses if negative; space otherwise; located flush left
[	]	Parentheses if negative; nothing otherwise
[>	>]	Parentheses if negative; nothing otherwise; located flush left (prefix); fixed position (suffix)
<[>		Parentheses if negative; nothing otherwise; located flush left
*	*	Extendable number of digits to left (prefix) or to right (suffix)
_	_	Space
^	^	Fill character (defined in options)
	th	Ordinality of number (st, nd, rd, or th)
	TH	Ordinality of number (ST, ND, RD, or TH)

Options: The following options are available:

Option	Explanation
uc	Make uppercase
lc	Make lowercase
left	Left aligned
right	Right aligned
ro	Read (XML) only; no editing allowed
edit	The field is editable (active by default)
dec=<char>	Specify a character for the decimal point (default is point)
fill=<char>	Specify fill character
base=<year>	Base year for year fields (see Note below)
pos	Show only positive numbers; input of negative numbers allowed

Note on Base Year: When using short year formats (for example, `DD-MM-YY{base=1940}`), the base year specifies a cut-off for a century. If the user enters a value that is equal to or greater than the last two digits (considered together as a number) of the base year, then the century is the same as that of the base year. If the user value is less than the last two digits, the century is the century of the base year plus one. For example if you set `base=1940`, then if the <DOCED> user enters "50", the XML value for the year will be `1950`; if the user enters "23", the XML value will be `2023`. Although two digits are commonly used as the short year format, one-digit and three-digit short year formats can also be used with a base year. Note that if you use the `y` field type, the base year is taken to be 1930.

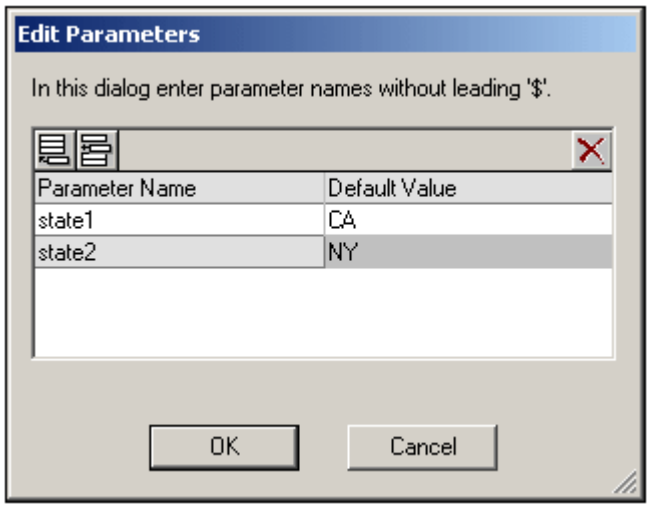
© 2005 Altova GmbH

4.13 Parameters

In StyleVision, you can declare parameters globally for the entire StyleVision Power Stylesheet. A parameter is declared with a name and a string value. This value is the default value and can be overridden by a value passed from the command line.

Mechanism

1. You declare one or more global parameters using the **Edit | Edit Stylesheet Parameters...** command. This pops up the Edit Parameters dialog (shown below). Declaring a parameter involves giving it a name and a string value, which is its default value. If no value is specified, the default value is an empty string.



   In the Edit Parameters dialog, you can append or insert new parameters, as well as edit existing parameters. Parameter names **must begin with** a letter, and can contain the characters `A` to `Z`, `a` to `z`, `0` to `9`, and the underscore.
2. You then define where in the StyleVision Power Stylesheet a parameter is to be used. You use parameters by calling them in XPath expressions with **`$paramname`**. For instance, you could call a parameter in the XPath of an Auto-Calculation (e.g. `concat('www.', $company, '.com')`).
3. You can pass a parameter in the stylesheet a value from the command line (see StyleVision from the command line for details). This value overrides the default value of the parameter.

Use of parameters
Parameters are useful in the following situations:

- If you wish to use one value in multiple locations. In this case, you can save the required value as a parameter value and use the parameter in all the required locations.
- If you wish to pass a value to the stylesheet at processing time. In the stylesheet, you use a parameter with a default value. At processing time, you pass the desired value to the parameter via the command line.

Note:

- Using parameters as described above allows you to declare parameters with default values in the StyleVision Power Stylesheet. When you generate the XSLT stylesheets, parameters will be included in these. You can then use the command line to generate HTML output. Parameters are currently not supported in Authentic View. This means that parameters used in the StyleVision Power Stylesheet will **not** be processed for the Authentic View output.

- The Edit Parameter dialog contains **all** the parameters that have been defined for the stylesheet.
- While it is an error to use an undeclared parameter in the StyleVision Power Stylesheet, it is not an error to declare a parameter and not use it.

© 2005 Altova GmbH

4.14 Unparsed Entity URIs

If you are using a DTD and have declared an unparsed entity in it, you can use the XSLT function `unparsed-entity-uri` to pass the URI of an unparsed entity from the DTD to the output. In StyleVision, you can use the `unparsed-entity-uri` function for the dynamic addresses of images and hyperlinks. To use this function, do the following:

1. Define an entity in the DTD as unparsed and associate a URI with it.

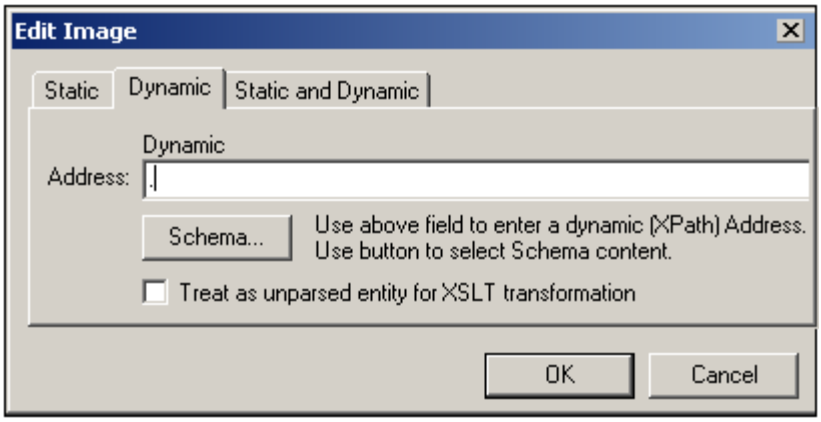
   ```
   <!ENTITY picture SYSTEM "nanonull.gif" NDATA GIF>
   ```

2. The entity name can then be used in an XSLT stylesheet. For example:

   ```
   <image source="picture" />
   ```

   When the stylesheet is processed, the URI associated with the entity name is substituted for the entity name.

3. IN StyleVision, the option of treating a node value as the name of an unparsed entity is available in the Image Properties and Hyperlink Properties dialogs.



When the node selected in the dynamic part of an image or hyperlink address contains the name of an unparsed entity, selecting the **Treat as unparsed entity** option causes the node value to be read as the argument of the `unparsed-entity-uri` function. For example, when the context node is `image/@source` and the Treat as Unparsed Entity option is checked, then the XSLT generated for this construct will be:

```
<xsl:template match="//image/@source>
  <img>
    <xsl:attribute name="src">
      <xsl:value-of select="unparsed-entity-uri(.)" />
    </xsl:attribute>
  </img>
</xsl:template>
```

When the XML document is processed, this XSLT fragment generates the URI associated with the unparsed entity named in the `source` attribute of `image` elements. If the URI is a relative URI, then the XSLT processor expands it to an absolute URI applying the base URI of the DTD. So the above XSLT fragment outputs something like:

```
file:///c:/some_folder/nanonull.gif
```

where the DTD is in the folder `some_folder`.

4.15 StyleVision from the command line

The command line feature enables you to call StyleVision (i.e. the StyleVision executable file) from the command line to generate the required output files. StyleVision runs silently (i.e. without the GUI being opened), generates the required output files, and closes.

Output files
Using the command line, you can generate one or more of the following files:

- XSLT directly from the specified StyleVision Power Stylesheet
- HTML file using the XML and XSLT specified in the StyleVision Power Stylesheet or using alternative XML and/or XSLT

Syntax
The command line syntax is as follows (names that can vary are indicated in italic; optional arguments are enclosed in square brackets):

`StyleVisionBatch [Stylevision executable] [`*`file`*`.sps] [parameter declarations] [options]`

`StyleVisionBatch`	located in the StyleVision application folder
`Stylevision exe`	specify only if your StyleVision executable is **not** `stylevision.exe` in the same folder
`file``.sps`	the StyleVision Power Stylesheet (`.sps` file) to open
`parameters`	parameters for the XSLT; must have the form `$paramname=paramvalue`. Multiple parameters must be separated with spaces

Options:

`/InpXML=`*`file`*	set input XML file
`/InpXSLT=`*`file`*	set input XSLT-for-HTML file
`/OutXSLT=`*`file`*	output stylesheet for HTML
`/OutHTML=`*`file`*	output HTML
`/help` or `/?`	outputs help at the command line
`/verbose` or `/v`	reports all steps in the process at the command line

Examples

```
StyleVisionBatch examples\file.sps /outhtml=examples\file.html
StyleVisionBatch /inpxml=source.xml /inpxslt=mystyle.xslt
   /outhtml=examples\file.html
```

Note:

- When `StyleVisionBatch` is called, it looks in the current directory for `stylevision.exe`. If your StyleVision executable is named otherwise or located in another folder, use the option to specify the executable.
- Paths may be absolute or relative and should use backslashes.
- Commands, paths, and folder and file names are case-insensitive.
- If the SPS file is specified, the Working XML File associated with it and XSLT generated from it will be used to generate output; therefore no input XML or XSLT file is required. If, however, the SPS file is not specified, an input XML file and input XSLT file must be specified as options. An input XML File must also be specified if the SPS file does not have a Working XML File assigned it.
- Parameter declarations refer to parameters in the XSLT. Parameter names and values are case-sensitive. Each parameter declaration on the command line must be prefixed

 © 2005 Altova GmbH

with a `$`, and, if multiple parameters are used, they must be separated from each other with a space.

- No default output is specified, so you must specify the required output.
- If you specify only the output file (no XML or XSLT), the Working XML File specified in the StyleVision Power Stylesheet is used for the source XML and the required XSLT is generated from the StyleVision Power Stylesheet (SPS file). Any temporary files that are created are deleted at the end of the processing.
- The `/verbose` option provides a detailed report of all steps carried out during the processing of the command.
- When specifying HTML output, make sure that the generated file is placed in a location in which relative paths to images, etc, will point correctly to their targets.

4.16 Authentic View

In StyleVision, you can preview the Authentic View of an XML document that has been assigned as the Working XML File to the currently open Altova StyleVision 2005. This preview is available when you click the Authentic Preview tab in the Main Window of StyleVision. In order for the Authentic Preview to be enabled, you must assign the currently open StyleVision Power Stylesheet a Working XML File (**File | Assign Working XML File**).

This section provides:

- An overview of the interface
- A description of the toolbar icons specific to Authentic View
- A description of viewing modes available in the main Authentic Preview window
- A description of the context menus available at various points in the Authentic View of the XML document
- A detailed description of how to use various Authentic View features

Additional sources of Authentic View information are:

- An Authentic View Tutorial, which shows you how to use the Authentic View interface. This tutorial is available in the documentation of the Altova XMLSpy 2005 and Altova Authentic 2005 Desktop Edition products (see the Tutorials section), as well as online.
- For a detailed description of Authentic View menu commands, see the User Reference section of your product documentation.

© 2005 Altova GmbH

4.16.1 Overview of the GUI

The Authentic Preview provides you with menu commands, toolbar icons, and context menus with which to edit the XML document that is displayed in the Main Window.

Menu bar
The menus available in the menu bar are described in detail in the User Reference section of your product documentation.

Toolbar
The symbols and icons displayed in the toolbar are described in the section, Authentic View toolbar icons.

Main window
This is the window in which the Working XML document is displayed and edited. It is described in the section, Authentic View main window.

Status Bar
The Status Bar displays the XPath to the currently selected node. In the Authentic Preview of StyleVision, the XPath to the currently selected node is indicated in the Schema Tree, where the currently selected node is highlighted in gray. The XPath in Authentic Preview is not displayed in a status bar.

Context menus
These are the menus that appear when you right-click in the Main Window. The available commands are context-sensitive editing commands, i.e. they allow you to manipulate structure and content relevant to the selected node. Such manipulations include inserting, appending, or deleting a node, adding entities, or cutting and pasting content.

4.16.2 Authentic View toolbar icons

Icons in the Authentic View toolbar are command shortcuts. Some icons will be already familiar to you from other Windows applications or your Altova product, others might be new to you. This section describes icons unique to Authentic View.

In the description below, related icons are grouped together.

Show/hide XML markup

Markup tags can be turned on or off in Authentic Preview.

 Hide markup.

 Show all markup. XML element/attribute tags are shown with names.

Editing dynamic table structures

Rows in a **dynamic SPS table** are repetitions of a data structure. Each row represents an occurrence of a single element. Each row, therefore, has the same XML substructure as the next.

The dynamic table editing commands manipulate the rows of a dynamic SPS table. That is, you can modify the number and order of the element occurrences. You cannot, however, edit the columns of a dynamic SPS table, since this would entail changing the substructure of individual element occurrences.

The icons for dynamic table editing commands appear in the toolbar, and are also available in the **Authentic** menu.



 Append row to table

 Insert row in table

 Duplicate current table row (i.e. cell contents are duplicated)

 Move current row up by one row

 Move current row down by one row

 Deletes the current row

Note: These commands apply only to **dynamic SPS tables**. They should not be used inside static SPS tables.

Define Entities icon

 This icon opens the Define Entities dialog, which allows the Authentic View user to define entities that can then be used in the XML document.

© 2005 Altova GmbH

Save XML icon

 Saves changes made in Authentic Preview to the Working XML File.

4.16.3 Authentic View main window

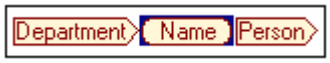
There are four viewing modes in Authentic View: Large Markup; Small Markup; Mixed Markup; and Hide All Markup. These modes enable you to view the document with varying levels of markup information. In Authentic Preview of StyleVision only two markup modes are available: Hide Markup and Show Large (Full) Markup.

To switch between modes, use the commands in the Authentic menu or the icons in the toolbar (see the previous section, Authentic View toolbar icons).

Large markup
This shows the start and end tags of elements and attributes with the element/attribute names in the tags:

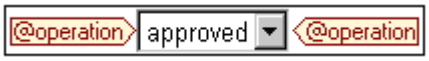


The element `Name` in the figure above is **expanded**, i.e. the start and end tags, as well as the content of the element, are shown. An element/attribute can be **contracted** by double-clicking either its start or end tag:



To expand the contracted element/attribute, double-click the contracted tag.

In large markup, attributes are recognized by the symbol `@` in the start and end tags of the attribute:



Hide all markup
All XML markup is hidden. Since the formatting seen in Authentic View is the formatting of the printed document, this viewing mode is a WYSIWYG view of the document.

Content display
In Authentic View, content is displayed in two ways:

- Plain text. You type in the text, and this text becomes the content of the element or the value of the attribute.

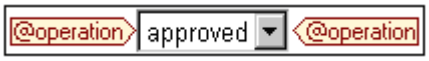
  

- Data-entry devices. The display contains either an input field (text box), a multiline input field, combo box, check box, or radio button. In the case of input fields and multiline input fields, the text you enter in the field becomes the XML content of the element or the value of the attribute.

  In the case of the other data-entry devices, your selection produces a corresponding XML value, which is specified in the StyleVision Power Stylesheet. Thus the selection "approved" in the display example below could map to an XML value of "1", or to "approved", or anything else; while "not approved" in the display could map to "0", or "not approved", or anything else.

 © 2005 Altova GmbH



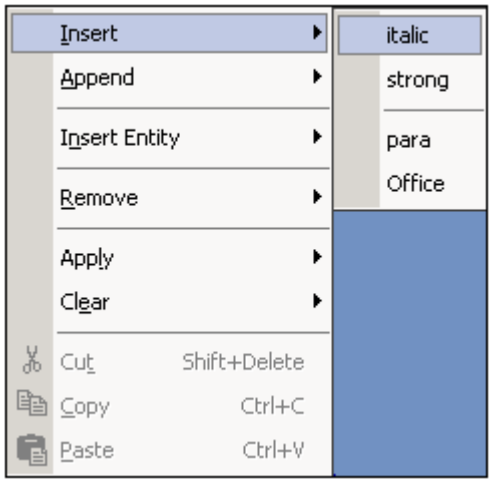
Optional nodes
When an element or attribute is **optional** (according to the referenced schema), a prompt of type "add [*element/attribute*]" is displayed:



Clicking the prompt, adds the element, and places the cursor for data entry. If there are multiple optional nodes, the prompt "add..." is displayed. Clicking the prompt, displays a menu of the optional nodes.

4.16.4 Authentic View context menus

Right-clicking on some selected document content or node pops up a menu with commands relevant to the selection or cursor location. This kind of menu is called a context menu. The context menu that appears depends on where in the document the cursor is. The context menu, with applicable commands enabled, is shown below. The figure below also shows the Insert submenu, which is a list of all elements that can be inserted at that point. Note that the **Insert submenu** shows (above the rule) the elements that can be **inserted within the current element**, and (below the rule) the elements that can be **inserted before the current element**. In the figure below, the current element is the `para` element. The `italic` and `strong` elements can be inserted within the current `para` element. The `para` and `Office` elements can be inserted before the current `para` element.



Most of the commands available in the context menu are explained in Authentic View entry helpers.

Remove element
Pointing over the **Remove** command pops up a menu list consisting of the selected element and all its ancestors up to the document element. Click the element to be removed. This is a quick way to delete an element or any of its ancestors. Note that clicking an ancestor element will remove all its descendants, including the selected element.

© 2005 Altova GmbH

4.16.5 Features and their usage

This section describes important features of Authentic View in detail. Features have been included in this section either because they are commonly used or require an explanation of the mechanisms or concepts involved.

The section explains the following:

- The Date Picker is a graphical calendar that enters dates in the correct XML format when you click a date. See Using the Date Picker.
- An entity is shorthand for a special character or text string. You can define your own entities, which allows you to insert these special characters or text strings by inserting the corresponding entities. See Define Entities for details.
- What image formats can be displayed in Authentic View.

To learn how to use all the features of Authentic View, please do the Authentic View Tutorial using either XMLSpy 2005 or Authentic 2005 Desktop Edition. The Authentic View Tutorial is available with these products.

Tables in Authentic View

The three table types fall into two categories: SPS tables (static and dynamic) and XML tables.

SPS tables are of two types: static and dynamic. SPS tables are designed by the designer of the StyleVision Power Stylesheet to which your XML document is linked. You yourself cannot insert an SPS table into the XML document, but you can enter data into SPS table fields and add and delete the rows of dynamic SPS tables. The section on SPS tables below explains the features of these tables.

XML tables are inserted by you, the user of Authentic View. Their purpose is to enable you to insert tables at any allowed location in the document hierarchy should you wish to do so. The editing features of XML tables and the XML table editing icons are described below.

SPS Tables

Two types of SPS tables are used in Authentic View: static tables and dynamic tables.

Static tables are fixed in their structure and in the content-type of cells. You, as the user of Authentic View, can enter data into the table cells but you cannot change the structure of these tables (i.e. add rows or columns, etc) or change the content-type of a cell. You enter data either by typing in text, or by selecting from options presented in the form of check-box or radio button alternatives or as a list in a combo-box. After you enter data, you can edit it.

Nanonull, Inc.

Street:	119 Oakstreet, Suite 4876	**Phone:**	+1 (321) 555 5155
City:	Vereno	**Fax:**	+1 (321) 555 5155 - 9
State & Zip:	DC 29213	**E-mail:**	office@nanonull.com

Note: The icons or commands for editing dynamic tables **must not** be used to edit static tables.

Dynamic tables have rows that represent a repeating data structure, i.e. each row has an identical data structure (not the case with static tables). Therefore, you can perform row operations: append row, insert row, move row up, move row down, and delete row. These commands are available under the **Authentic** menu and as icons in the toolbar (shown below).



To use these commands, place the cursor anywhere in the appropriate row, and then select the required command.

 © 2005 Altova GmbH

Administration

First	Last	Title	Ext	EMail	Shares	Leave		
						Total	Used	Left
Vernon	**Callaby**	Office Manager	581	v.callaby@nanonull.com	1500	25	4	21
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	0	22	2	20
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	add Shares	25	7	18
Employees: 3 (20% of Office, 9% of Company)					**Shares: 1500 (13% of Office, 6% of Company)**			
Non-Shareholders: Frank Further, Loby Matise.								

To move among cells in the table, use the Up, Down, Left, and Right arrow keys. To move forward from one cell to the next, use the Tab key. Pressing the Tab key in the last cell of a row creates a new row.

XML Tables

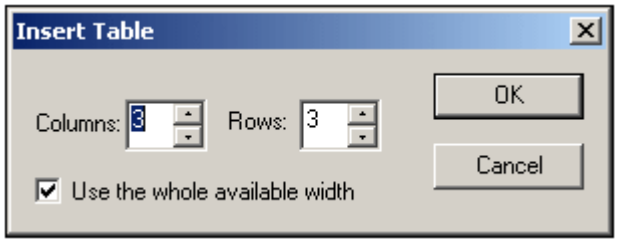
XML tables can be inserted by you, the user of Authentic View. They enable you to insert tables anywhere in the XML document where they are allowed, which is useful if you need to insert tabular information in your document. These tables will be printed out as tables when you print out directly from Authentic View. If you are also generating output with XSLT stylesheets, discuss the required output with the designer of the StyleVision Power Stylesheet.

Note that you can insert XML tables only at allowed locations. These locations are specified in the schema (DTD or XML Schema). If you wish to insert a table at additional locations, discuss this with the person designing the StyleVision Power Stylesheet.

Working with XML tables
There are three steps involved when working with XML tables: inserting the table; formatting it; and entering data. The commands for working with XML tables are available as icons in the toolbar (see XML table editing icons). Currently, XML tables cannot be inserted in the Authentic Preview of StyleVision.

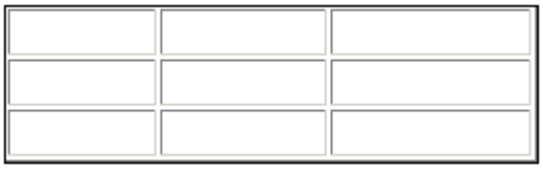
Inserting tables

To insert an XML table place your cursor where you wish to insert the table, and click the  icon. (Note that where you can insert tables is determined by the schema.) This pops up the Insert Table dialog (shown below).



Select the number of columns and rows, and specify whether you wish the table to extend the

entire available width. For the specifications given in the dialog box shown above, the following table is created.

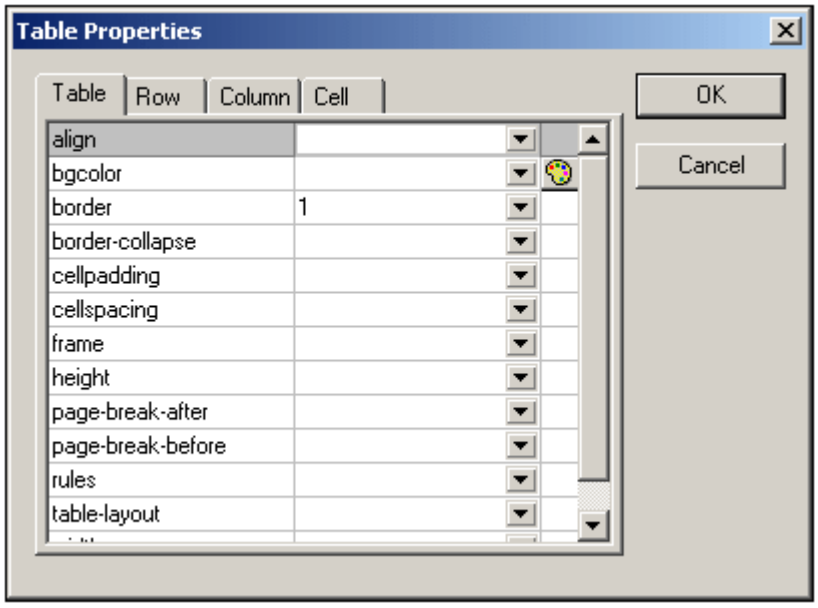


You can add and delete columns, create row and column joins later. So create the broad structure first.

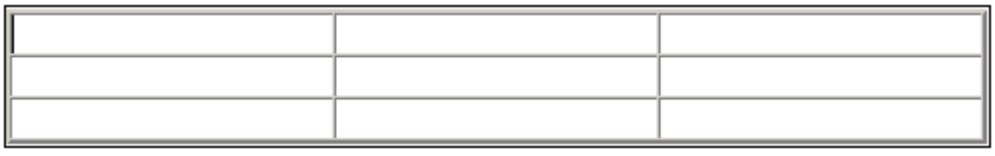
Note: All modifications to table structure must be made by using the Table menu commands. They cannot be made by changing attribute values in the Attribute Entry Helper.

Formatting tables and entering data

To format your table, place the cursor anywhere in the table and click the  (Table Properties) icon. This pops up the Table Properties dialog (shown below), where you specify formatting for the table, or for a row, column, or cell.

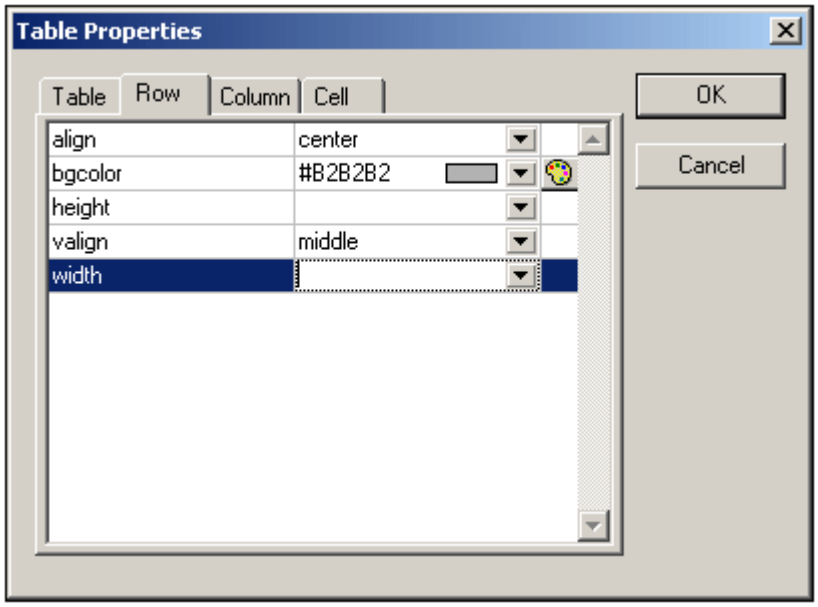


Set the cellspacing and cellpadding properties to "0". Your table will now look like this:



Now place the cursor in the first row to format it, and click the  (Table Properties) icon. Click the Row tab.

 © 2005 Altova GmbH



Since the first row will be the header row, set a background color to differentiate this row from the other rows. Note the Row properties that have been set in the figure above. Then enter the column header text. Your table will now look like this:

Name	Telephone	Email

Notice that the alignment is centered as specified. Now, say you want to divide the "Telephone" column into the sub-columns "Office" and "Home", you would need to join cells. Place the cursor in the "Telephone" cell, and click the  (Split vertically) icon. Your table will look like this:

Name	Telephone	Email

Now place the cursor in the cell below the cell containing "Telephone", and click the  (Split horizontally) icon. Then type in the column headers "Office" and "Home". Your table will now look like this:

Name	Telephone		Email
	Office	Home	

Now you will have to vertically split each cell in the "Telephone" column.

You can also add and delete columns and rows, and vertically align cell content, using the table-editing icons. The XML table editing icons are described in the User Reference, in the section

titled "XML Table Icons".

Moving among cells in the table
To move among cells in the XML table, use the Up, Down, Right, and Left arrow keys.

Entering data in a cell
To enter data in a cell, place the cursor in the cell, and type in the data.

Formatting text
Text in an XML table, as with other text in the XML document, must be formatted using XML elements or attributes. To add an element, highlight the text and double-click the required element in the Elements Entry Helper. To specify an attribute value, place the cursor within the text fragment and enter the required attribute value in the Attributes Entry Helper. After formatting the header text bold, your table will look like this.

Name	Telephone		Email
	Office	Home	

The text above was formatted by highlighting the text, and double-clicking the element `strong`, for which a global template exists that specifies bold as the font-weight. The text formatting becomes immediately visible.

Note: For text formatting to be displayed in Authentic View, a global template with the required text formatting must have been created in StyleVision for the element in question.

XML Table Editing Icons

The commands required to edit XML tables are available as icons in the toolbar, and are listed below. Note that no corresponding menu commands exist for these icons and that they are not active in StyleVision.

For a full description of when and how XML tables are to be used, see XML tables.

Insert table



The "Insert Table" command inserts a **CALS / HTML table** at the current cursor position.

Delete table



The "Delete table" command deletes the currently active table.

Append row



The "Append row" command appends a row to the end of the currently active table.

Append column



The "Append column" command appends a column to the end of the currently active table.

© 2005 Altova GmbH

Insert row



The "Insert row" command inserts a row above the current cursor position in the currently active table.

Insert column



The "Insert column" command inserts a column to the left of the current cursor position in the currently active table.

Join cell left



The "Join cell left" command joins the current cell (current cursor position) with the cell to the left. The tags of both cells remain in the new cell, the column headers remain unchanged.

Join cell right



The "Join cell right" command joins the current cell (current cursor position) with the cell to the right. The tags of both cells remain in the new cell, the column headers remain unchanged.

Join cell below



The "Join cell below" command joins the current cell (current cursor position) with the cell below. The tags of both cells remain in the new cell, the column headers remain unchanged.

Join cell above



The "Join cell above" command joins the current cell (current cursor position) with the cell above. The tags of both cells remain in the new cell, the column headers remain unchanged.

Split cell horizontally



The "Split cell Horizontally" command creates a new cell to the right of the currently active cell. The size of both cells, is now the same as the original cell.

Split cell vertically



The "Split cell Vertically" command creates a new cell below the currently active cell.

Align top



This command aligns the cell contents to the top of the cell.

Center vertically



This command centers the cell contents.

Align bottom

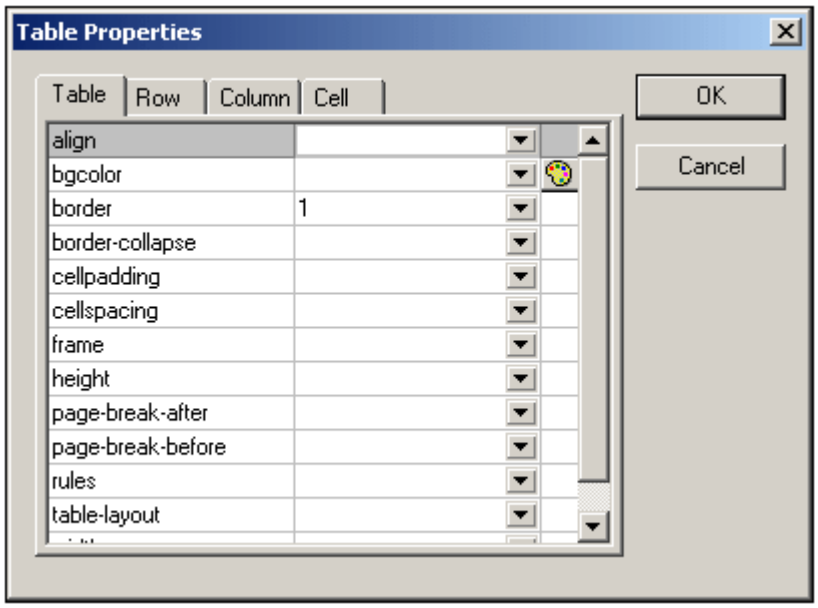


This command aligns the cell contents to the bottom of the cell.

Table properties



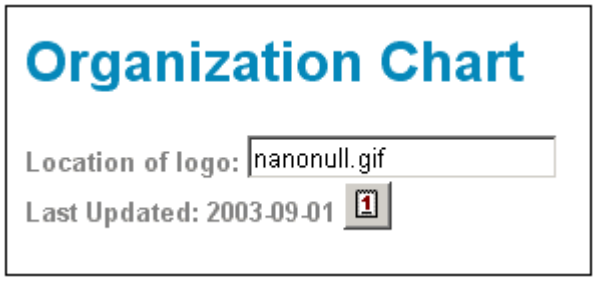
The "Table properties" command opens the Table Properties dialog box. This icon is only made active for HTML tables, it cannot be clicked for CALS tables.



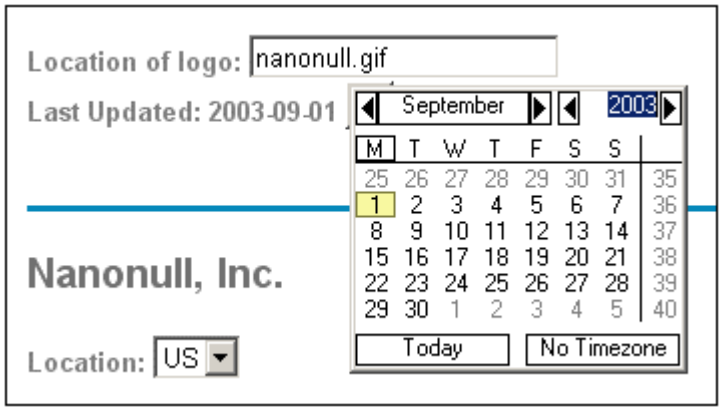
 © 2005 Altova GmbH

Date Picker

The Date Picker is a graphical calendar used to enter dates in a standard format into the XML document. Having a standard format is important for the processing of data in the document. The Date Picker icon appears near the date field it modifies; it is shown below.



To display the Date Picker (shown below), click the Date Picker icon.



To select a date, click on the desired date, month, or year. The date is entered in the XML document, and the date in the display is modified accordingly. You can also enter a time zone if this is required.

Define Entities

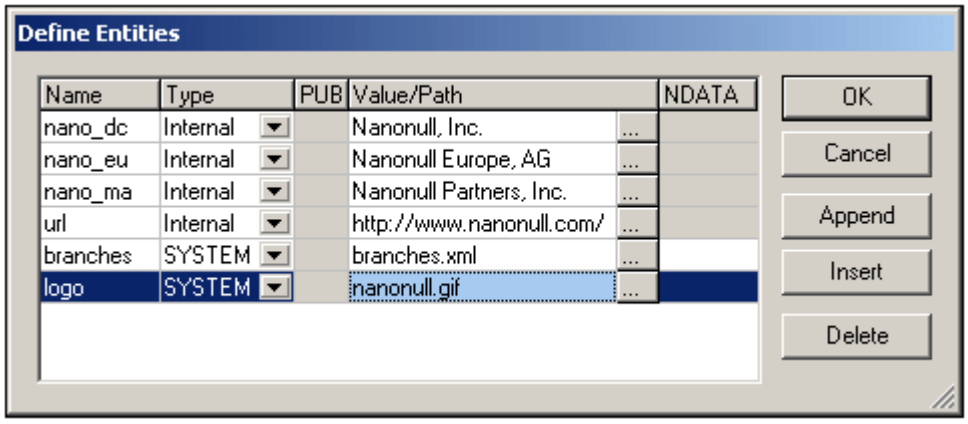
You can define entities for use in Authentic View, whether your document is based on a DTD or an XML Schema. Once defined, these entities are displayed in the Entities Entry Helper and in the **Insert Entity** submenu of the context menu. When you double-click on an entity in the Entities Entry Helper, that entity is inserted at the cursor insertion point.

An entity is useful if you will be using a text string, XML fragment, or some other external resource in multiple locations in your document. You define the entity, which is basically a short name that stands in for the required data, in the Define Entities dialog. After defining an entity you can use it at multiple locations in your document. This helps you save time and greatly enhances maintenance.

There are two broad types of entities you can use in your document: a **parsed entity**, which is XML data (either a text string or a fragment of an XML document), or an **unparsed entity**, which is non-XML data such as a binary file (usually a graphic, sound, or multimedia object). Each entity has a name and a value. In the case of parsed entities the entity is a placeholder for the XML data. The value of the entity is either the XML data itself or a URI that points to a `.xml` file that contains the XML data. In the case of unparsed entities, the value of the entity is a URI that points to the non-XML data file.

To define an entity, do the following:

1. Click **Authentic | Define Entities...**. This opens the Define Entities dialog.



2. Enter the name of your entity in the Name field. This is the name that will appear in the Entities Entry Helper.
3. Enter the type of entity from the drop-down list in the Type field. Three types are possible. An **Internal** entity is one for which the text to be used is stored in the XML document itself. Selecting **PUBLIC** or **SYSTEM** specifies that the resource is located outside the XML file, and will be located with the use of a public identifier or a system identifier, respectively. A system identifier is a URI that gives the location of the resource. A public identifier is a location-independent identifier, which enables some processors to identify the resource. If you specify both a public and system identifier, the public identifier resolves to the system identifier, and the system identifier is used.
4. If you have selected PUBLIC as the Type, enter the public identifier of your resource in the PUBLIC field. If you have selected Internal or SYSTEM as your Type, the PUBLIC field is disabled.
5. In the Value/Path field, you can enter any one of the following:

   - If the entity type is Internal, enter the text string you want as the value of your entity. Do not enter quotes to delimit the entry. Any quotes that you enter will be treated as part of the text string. Note that entities are a good mechanism for including Unicode characters in your document; do this by entering the Unicode number as the value of

 © 2005 Altova GmbH

an internal entity.
- If the entity type is SYSTEM, enter the URI of the resource or select a resource on your local network by using the Browse button. If the resource contains parsed data, it must be an XML file (i.e. it must have a `.xml` extension). Alternatively, the resource can be a binary file, such as a GIF file.
- If the entity type is PUBLIC, you must additionally enter a system identifier in this field.

6. The NDATA entry tells the processor that this entity is not to be parsed but to be sent to the appropriate processor. The NDATA field should therefore be used with unparsed entities only.

Dialog features
You can append, insert, and delete entities by clicking the appropriate buttons. You can also sort entities on the alphabetical value of any column by clicking the column header; clicking once sorts in ascending order, twice in descending order. You can also resize the dialog box and the width of columns.

Limitations

- An entity contained within another entity is not resolved, either in the dialog, Authentic View, or XSLT output, and the ampersand character of such an entity is displayed in its escaped form, i.e. `&`.
- External entities are not resolved in Authentic View, except in the case where an entity is an image file and it is entered as the value of an attribute of type `ENTITY` or `ENTITIES`. Such entities are resolved when the document is processed with an XSLT generated from the StyleVision Power Stylesheet.

Images in Authentic View

Authentic View is based on Internet Explorer, and is able to display most of the image formats that your version of Internet Explorer can display. The following commonly used image formats are supported:

- GIF
- JPG
- PNG
- BMP
- WMF (Microsoft Windows Metafile)
- EMF (Enhanced Metafile)
- SVG (for PDF output only)

© 2005 Altova GmbH

Keystrokes in Authentic View

Enter (Carriage Return) Key
In Authentic View the **Return** key is used to append additional elements when it is in certain cursor locations. For example, if the chapter of a book may (according to the schema) contain several paragraphs, then pressing **Return** inside the text of the paragraph causes a new paragraph to be appended immediately after the current paragraph. If a chapter can contain one title and several chapters, pressing Enter inside the chapter but outside any paragraph element (including within the title element) causes a new chapter to be appended after the current chapter (assuming that multiple chapters are allowed by the schema).

Note: The **Return** key does **not** insert a carriage return/line feed, i.e. it does not jump to a new line. This is the case even when the cursor is inside a text node, such as paragraph.

Chapter 5

Tips

5 Tips

This section lists tips that enable you to use StyleVision features to advantage. It also points out some common errors. The section is divided into sub-sections according to function. Tips and commonly made errors for each heading are discussed together. For more information about issues raised in this section, refer to the relevant sections in the StyleVision How To Use section.

© 2005 Altova GmbH

5.1 The schema for the StyleVision Power Stylesheet

XML Schema and the Root Template
If you use an XML Schema (and not a DTD) as your schema, you will very likely have declared more than one global element. In such a case it is not possible for StyleVision to determine which one of these global elements it should consider the document element (top-level element). Consequently, the Root Template will be empty and the global elements will only appear in the Global Templates list. In order to enable StyleVision to select a document element, assign a Working XML File (**File | Assign Working XML File**). The document element of the Working XML File will then be used as the document element in the Root Template.

Modifying your schema
It is best to finalize your schema before you start building a StyleVision Power Stylesheet (SPS). Sometimes, however, you may not be able to avoid modifying your schema after having built the StyleVision Power Stylesheet. If you do change your schema after building the StyleVision Power Stylesheet, the following effects must be considered:

- When an XML file is created or edited in Authentic View with a StyleVision Power Stylesheet that references a modified schema, then Authentic View will validate the XML file against the modified schema. If the templates in the StyleVision Power Stylesheet are still valid according to the modified schema, then the XML that is created will be valid and there should be no problem. If, however, a template in the StyleVision Power Stylesheet uses an element or structure which is not valid according to the modified schema, then an error results. To prevent such an error from occurring, you should change the StyleVision Power Stylesheet so that it is valid according to the modified schema (see next point).
- When a StyleVision Power Stylesheet based on a modified schema is opened in StyleVision, the modified schema appears in the Schema Window and any template that uses elements or structures which have been modified may now be incorrect. StyleVision will indicate this. To rectify this situation, you have two options:
    1. If the change to the element or structure is minor, or if the StyleVision Power Stylesheet does not use complex template rules, you could change the incorrect template rules.
    2. If the schema modifications are too complex or too many, or if the StyleVision Power Stylesheet uses complex template rules, it is preferable—and may even be necessary—to build the StyleVision Power Stylesheet from scratch all over again.

Renaming the schema or changing its location
If you rename your schema or change its location, you can open the StyleVision Power Stylesheet in XMLSpy 2005 or a plain-text editor, and change the value of the `schemafile` attribute of the `structure` element to specify the changed name or location of the schema file.

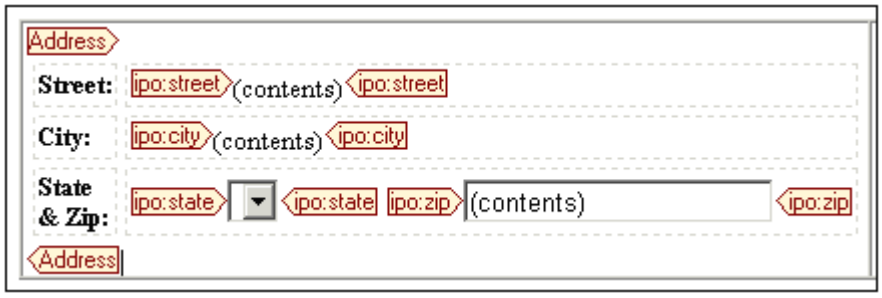
```
<structure version="2" schemafile="NanonullOrg.xsd"
   workingxmlfile="NanonullOrg.xml" templatexmlfile="">
```

Importing schemas
The schema on which the StyleVision Power Stylesheet is based can import other schemas. If you import schemas, pay attention to rules that govern the application of namespaces. See Namespaces in the How To section of this documentation.

5.2 Handling components in the Design Document

A component in the Design Document can be a node (including text nodes), data-entry device, placeholder, table, image, etc (see Document design process). Learning to handle components in the Design Document well will give you greater flexibility in using StyleVision and will enable you to create better StyleVision Power Stylesheets. Given below are a few tips for handling components.

Placing schema nodes correctly
When you drag nodes from the Schema Window and drop them into the Design Document, place them in a structure that is efficient. In the example below, a static table is created **within** the `Address` element tags because all content of the table cells are planned to be children of the `Address` element.



Do not create the table outside the `Address` element and then drag the children of `Address` into the table cells. In the example shown below, the context node for the static table is not the `Address` element but the parent of `Address`.



As a result, the `Address` element is repeated in the XPath expressions of each table-cell node. The processing of the nodes is slowed down and the Design Document becomes harder to read. Such constructs should, therefore, be avoided.

Moving components in the Design Document
You can move or copy any component from one part of the Design Document to another. You can do this using the following mechanisms:

- Drag-and-drop (move component)
- Cut-and-paste (move component)
- Copy-and-paste (copy component)

To move a component, first select it (either by clicking on it (in the case of nodes, for example) or by marking it (static text, for example)). Then, either drag it to the new location (move), or cut-and-paste it to the new location (move), or copy-and-paste it to the new location (copy).

Moving complex components, such as tables
You can even move or copy complex components such as a table and all its contents, a list, an auto-calculation, or a conditional template.

 © 2005 Altova GmbH

For example, to move a static table, mark the table from one whitespace before the table to one whitespace after the table, and drag (or cut/copy-and-paste) the static table to the new location. In the case of dynamic tables, select either tag of the element created as a table (these tags are located directly before and after the table), and drag (or cut/copy-and-paste) the selected table to the new location.

Moving dynamic content (nodes)
When you move or copy dynamic content, the XPath originally used to locate the node may be invalid in another location because the context node in the new location is different. In this case, an error message to this effect will appear, and the node that has been moved to the new context will have a red strikethrough in its tags:



To correct such an error, drag the correct context node to an adjacent location as shown here:



Then drag the descendant node into the context node, as below:



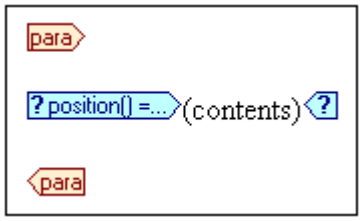
The red strikethrough disappears, indicating that the XPath in this location is now correct. Delete the `(contents)` placeholder of the context node to complete the operation (since you do not want its content to appear separately).

5.3 Selective processing of siblings

Function
An important requirement while selecting from among sibling elements of the same name is the ability to select by position. For example, you may wish to output only the first of several `para` elements (equivalent to the XPath `//para[1]`). In your StyleVision Power Stylesheet, the way to do this with XSLT 1.0 is by using a **conditional template** and the **`position()` function**. (Remember: a predicate of the form `//para[1]` is shorthand for `//para[position()=1]`.)

Usage
To select and process a specific sibling from among several of the same name, do the following:

1) Create the conditional template around the `(contents)` placeholder of the element you want to output.
2) The conditional template can occur anywhere in the document. However, you must ensure that the context node for the condition is the element for which you want to make the predicate. So if you wish to select the first `para` element, then the context node for the condition must be `para`. This means that the condition must be created directly within the `para` element tags.
3) For the condition, create an XPath expression that checks the position of the element, for example: `position()=1`. The Design Document will look like this:



The above steps do the following:

- For each `//para` element, check whether the position is 1.
- If the position is 1, then output the contents of this `para` element.
- If the position is not 1, then do not process this `para` element.

 © 2005 Altova GmbH

5.4 Manually editing generated XSLT

Occasionally, you may wish to manually edit your XSLT stylesheets. Typically this would be because you need some processing that is not currently available in StyleVision but is possible with some XSLT construct. In such cases, you can open the generated XSLT stylesheet in an XML editor and edit it as required.

However, note the following:

- If you manually edit a generated XSLT and then need to modify your StyleVision Power Stylesheet, then you will need to generate the XSLT again and re-do the manual edits.
- Manually editing the StyleVision Power Stylesheet itself to include XSLT features not supported by StyleVision will not enable these XSLT features in Authentic View. These features should only be inserted in the generated XSLT, which can then be used separately to transform an XML document.

Chapter 6

Troubleshooting

6 Troubleshooting

Given below is a list of commonly encountered problems, their causes, and suggested solutions.

- *When an XML Schema is loaded into StyleVision, the Root Template in the Schema Window is empty.*
  Cause: There is more than one global template in the schema. So it is not clear which one is to be used as the Root Element.
  Solution: If you have an XML file which has as its root element the element you require, assign this XML file as the Working XML File. Once you do this, it becomes clear which element is the root element, and the schema will appear in the Schema Window. If you do not have such an XML file, create a rudimentary one with the required element as the root element. Save it and assign it as the Working XML File. See The schema for the StyleVision Power Stylesheet.

- *No Global Templates are displayed in the Global Templates list in the Schema Window when the schema is loaded.*
  Cause: You are using an XML Schema (not a DTD), and there are no global elements in your XML Schema.
  Solution: If you wish to use global templates, you must have the required elements created as global elements in your XML Schema. (This is not a requirement for DTDs; all elements in a DTD are displayed in the Global Templates list.) In the XML Schema, create the required elements as global elements and reference them where they are required. See Global Templates.

- *The Block Style Window is disabled.*
  Cause: The currently selected component in the Design window is not a block-type component.
  Solution: All components in the Design Window are originally inline components. No component is inherently a block component. In order to make a component a block component, you have to explicitly specify a block-type formatting for the component. Do this by selecting the component in the Design Window and assigning it a Predefined Format (**Insert | Format...** or select an option in the Predefined Format combo box in the toolbar). All predefined formats except `Normal` are block type. See Formatting the components.

- *In the output, multiple paragraphs run on and are displayed together as a single paragraph instead of as multiple paragraphs.*
  Cause: In the Design Document, the node—as opposed to the `(contents)` placeholder—has been assigned the predefined `Paragraph (p)` format.
  Solution: Select the node and assign it a predefined format of `Normal`. Then select the `(contents)` placeholder and assign it a predefined format of `Paragraph (p)`. See Formatting the components.

- *An Auto-Calculation does not update the XML node specified in the Auto-Calculation dialog.*
  Cause: (1) The XPath specified for the Auto-Calculation does not use a value from the XML file and/or (2) none of the nodes used in the Auto-Calculation XPath has had its XML value modified. In other words, in order for a node in the XML file to be updated with the result of an Auto-Calculation, the XPath used in the Auto-Calculation (i) must use at least the value of one node in the XML file as an operand in the calculation, **and** (ii) at least one of those XML values (operated on in the Auto-Calculation XPath) must be modified in Authentic View.
  Solution: Force the Authentic View user to modify the value of a node used in the XPath for

 © 2005 Altova GmbH

the Auto-Calculation. You could do this explicitly with a message to the user (via **Authentic Node Settings**) or implicitly by creating an XPath for the Auto-Calculation that uses a node which must be updated. If you are using a static value as the XPath of your Auto-Calculation, such as a string value (for example, `string("www.altova.com")`), replace the string with a locator expression (such as `company/url`, where `url` is the node that contains the required value. However, the value of the XML node (e.g. `url`) will have to be modified in Authentic View for the Auto-Calculation function to pass the result to the XML node that is to be updated). See How to use Auto-Calculations.

- *XML tables do not display in HTML output.*
  Cause: No (global) template has been defined for the XML table element.
  Solution: Since it cannot be known where in the XML document the user will insert an XML table, you must use a global template for the XML table element. This global template must be added manually to the XSLT stylesheet you generate from the StyleVision Power Stylesheet.

- *In Authentic View, optional child elements are listed in a specific order (as `add child1`, `add child2`, etc) even though the schema specifies that child elements can occur in any order.*
  Cause: Templates for the child element/s have been specified within the template for the parent element.
  Solution: Delete the templates for the child element/s in the parent element. Define the template for the parent element to contain just the (contents) placeholder, and define Global Templates for the child elements.

Chapter 7

HTML to XML Conversion

7 HTML to XML Conversion

Altova StyleVision 2005 can generate a schema file (DTD or XML Schema), an XML file, and stylesheets from an HTML document.

In order to generate these files from the HTML document, you import the HTML document into the Design Window of StyleVision and use it to build a schema tree. In the Design Document view, you select the HTML elements or attributes you wish to include in the schema, and drag-and-drop them into the desired location in the schema tree. You can also add your own nodes to the schema tree, i.e. nodes that do not come from the HTML document. After you have created the schema tree, you can edit the Design Document as you would a normal StyleVision Power Stylesheet. At the end of the process, you will have created (i) a schema tree, and (ii) a StyleVision Power Stylesheet.

HTML-to-XML conversion steps
The HTML-to-XML conversion process consists of the following steps. Each step of the process is described in detail in the tutorial that follows this section.

1. Import the HTML file into StyleVision. The HTML will be displayed without markup tags in the Design Document view of StyleVision. At the same time a schema tree with a single element—a document element called `Root`—is created in the Schema Window.
2. Create the schema tree by (i) dragging nodes from the HTML document to the required location in the schema tree; and, optionally, (ii) adding your own nodes to the schema tree. In the Design Window, HTML content that has been used to build nodes in the schema tree will now be displayed with node tags around the content. HTML content that has no corresponding schema node will continue to be displayed without tags.
3. In the Design Document, assign formatting to nodes, refine processing rules, or add static content as required. These modifications will have an effect only on the StyleVision Power Stylesheet and the generated XSLT. It will not have an effect on either the generated schema or XML file.
4. After you have completed the schema tree and the design of the StyleVision Power Stylesheet, you can generate and save the following:
   - a schema (DTD or XML Schema) corresponding to the schema tree you have created;
   - an XML file based on the schema tree;
   - a StyleVision Power Stylesheet (`.sps` file) and/or XSLT stylesheet based on your Design Document.

Note: The generated schema and XML file are based on the schema tree you create. The StyleVision Power Stylesheet and XSLT are based on the Design Document.

 © 2005 Altova GmbH

7.1 HTML-to-XML Tutorial

In this tutorial, you will learn how to convert an HTML page into an XML document. You will do this by creating a schema, which StyleVision will then use to generate the XML document. You should note that the Design Document view of StyleVision, which displays a browser view of the source HTML document, is simultaneously a view of the StyleVision Power Stylesheet. You can save this StyleVision Power Stylesheet and also generate an XSLT from it.

In this tutorial, you will create and generate:

- An XML Schema: **article.xsd**
- An XML file based on this schema (with content from the HTML document): **article.xml**
- A StyleVision Power Stylesheet: **article.sps**
- An XSLT stylesheet: **article.xslt**

For this tutorial, you require the file **article.html**, which is in the **Altova/Altova StyleVision 2005/Examples** folder.

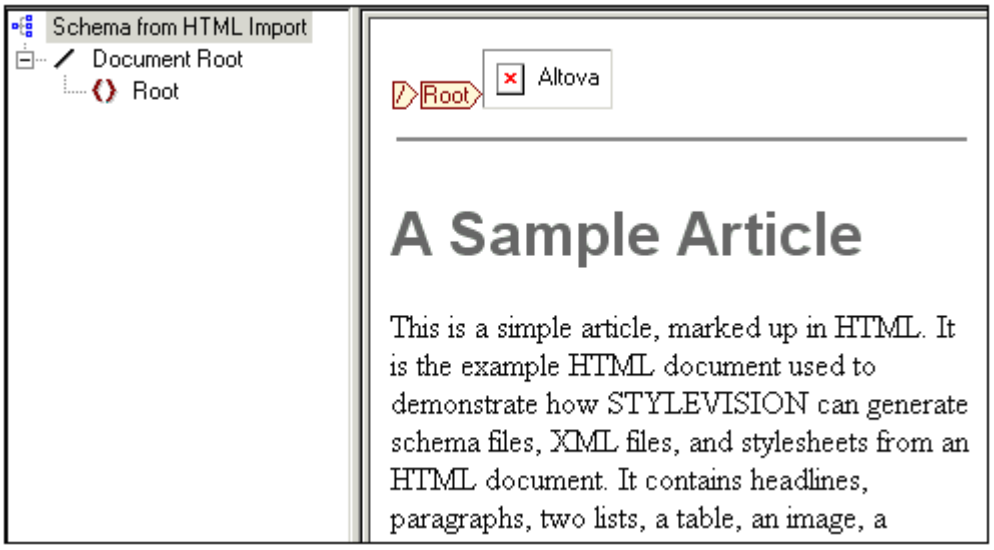
This tutorial consists of the following broad steps:

1. Open the file `article.html`.
2. Create the schema tree; this part forms the bulk of the tutorial.
3. Generate and save the output files.

If you need more details about StyleVision features that are used to create StyleVision Power Stylesheets, please see the StyleVision Tutorial and User Reference.

7.1.1 Opening the HTML

Open the file **article.html** in StyleVision by clicking **HTML Import | Import HTML file...** or **File | Import HTML file...** and browsing for article.html. This file is in the Altova/Altova StyleVision 2005/Examples folder. In StyleVision, article.html will be displayed in the Design Document window.


Schema from HTML Import
Document Root
Root
Root
Altova
A Sample Article
This is a simple article, marked up in HTML. It is the example HTML document used to demonstrate how STYLEVISION can generate schema files, XML files, and stylesheets from an HTML document. It contains headlines, paragraphs, two lists, a table, an image, a

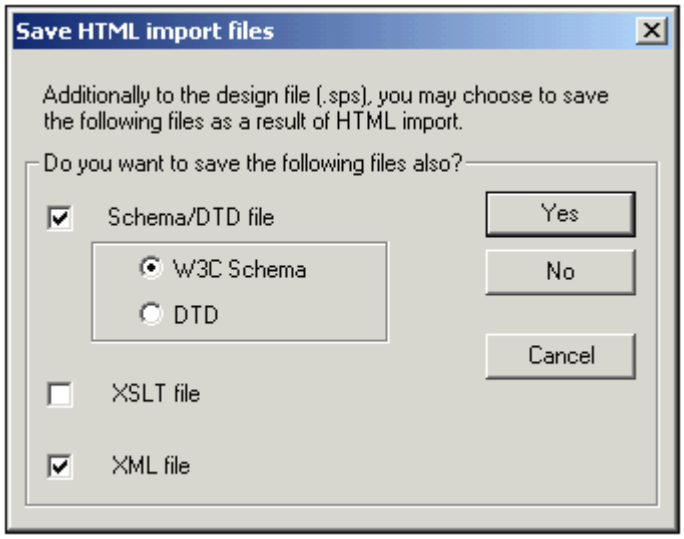
Note the following points:

- The HTML document is displayed without tags except for a **Root** tag.
- The schema tree in the Schema Window shows the Document Root with only one node, a document element called **Root**.
- There is only the Root Template. There are no global templates.

 © 2005 Altova GmbH

7.1.2 The starting schema and XML files

Now, without adding anything to the schema, generate and save the XML Schema and the XML files. Do this by clicking **File | Save Design**. The following dialog pops up:


Save HTML import files
Additionally to the design file (.sps), you may choose to save the following files as a result of HTML import.
Do you want to save the following files also?
Schema/DTD file
W3C Schema
DTD
XSLT file
XML file
Yes
No
Cancel

Make the selections as shown above, and click **Yes**. Save the XML Schema as `article.xsd` and the XML file as `article.xml`. You will also be asked whether you wish to save an SPS file. Do so, using the name `article.sps`. Now do the following:

- Open `article.xsd` in an XML editor like XMLSpy 2005 or in a text editor. You will see that it has only one element, named `Root`. This is because the schema tree in StyleVision from which it was generated contains only this one element.
- Now open `article.xml` in an editor program. The document has an empty `Root` element only. This is because `article.xsd`, on which the XML file is based, has just the one `Root` element.

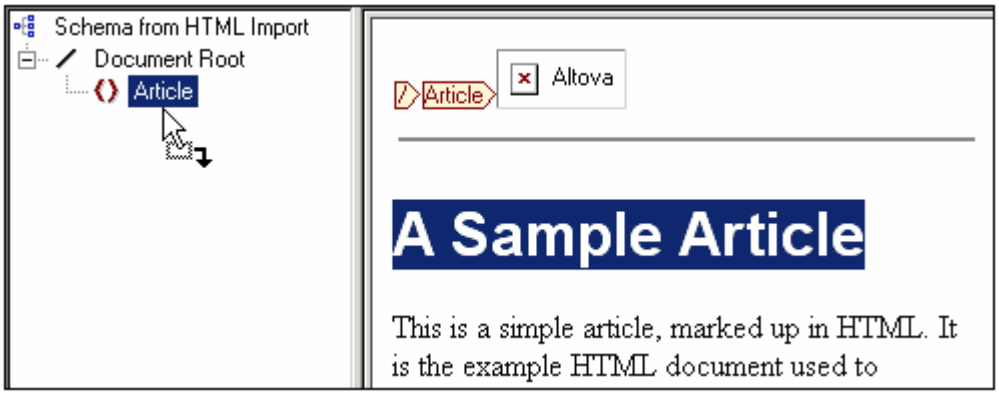
The important point to note is that the XML document is generated on the basis of the schema you create in StyleVision. In the next sections, you will learn how to create a schema that generates the required XML file.

7.1.3 Creating element nodes

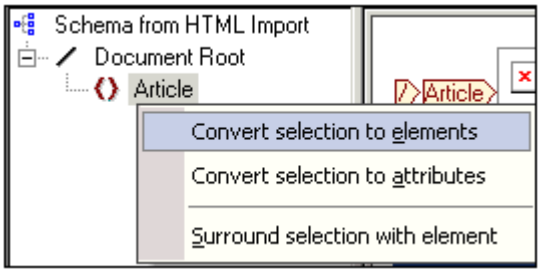
Renaming schema elements
Rename the element `Root` to `Article` by right-clicking `Root` in the schema tree, selecting **Rename**, and keying in the new name `Article`. Notice that the name changes in the both the schema tree as well as the Design Document.

Creating HTML content as an element

1. In the Design Document, select the article title "A Sample Article".
2. Drag the selection to the `Article` node in the schema tree till the pointer becomes a right-and-down arrow.


Schema from HTML Import
Document Root
Article
Article
Altova
A Sample Article
This is a simple article, marked up in HTML. It is the example HTML document used to

3. This indicates that the selected HTML content will be created as a child node (element or attribute) of `Article`; in other words, that a child node of `Article` will be created around the selected content. When you release the mouse button, the following menu pops up.



4. Select **Convert selection to elements** to create the selected HTML content as a child element node of `Article`.



5. In the schema tree, the new child node is created with a default name of `NewNode`. That it is created as contents (i.e. it has text content) is indicated by the (···) symbol. In the Design Document, the selected HTML content is converted to the familiar `(contents)`

placeholder of the StyleVision Power Stylesheet, and the start and end of the node is indicated with tags.

6. Right-click `NewNode` in the schema tree and rename it to `Title`. Notice that the node name in the tags in the Design Document also changes.

Note:

The following points should be noted:

- When you drag selected content into the Schema Window and prepare to drop, the type of arrow that appears indicates whether the dragged node can be created as a child (right-and-down arrow ) or a sibling (down arrow ).
- In the above example, you converted the selection to elements. This action changes the selected HTML content to a `(contents)` placeholder. This kind of a node is indicated in the schema tree by the  symbol. Alternatively, you could have selected **Surround selection with element**., which would have created an element node around the selected HTML content, but would have left the HTML content as static text (not dynamic text, indicated by the `(contents)` placeholder). Such an element node is indicated in the schema tree by the  symbol.

Now click **File | Save** to save the schema and XML files. If you look at the XML file, it should look something like this:

```
<Article xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="C:\workarea\html2xml\article.xsd">
   <Title>A Sample Article</Title>
</Article>
```

The document element is `Article` and it has a child `Title` element, which has the selected HTML content as its content.

Creating HTML content as an empty element node

If when creating the `Title` node in the schema tree, you had selected Surround selection with element, the `Title` node would appear in the XML document, **but it would be empty**; the selected HTML content would not be passed on to the XML document as the content of the `Title` element. You can try this out by doing the following:

1. In the schema tree, right-click `Title` and select Delete. This deletes the node in the schema tree, and restores the `(contents)` placeholder to its original text form: *A Sample Article*.
2. Select the title text, drag it to the schema tree so it will be a child of Article, and after dropping it select **Surround selection with element**.



Notice that instead of a `(contents)` placeholder being created, the original text remains and is surrounded by the start and end tags of the new node.

3. Rename `NewNode` to `Title`.
4. Save the schema and XML files.

If you look at the XML file, you will find that the `Title` element is empty.

```
<Article xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="C:\workarea\html2xml\article.xsd">
   <Title/>
</Article>
```

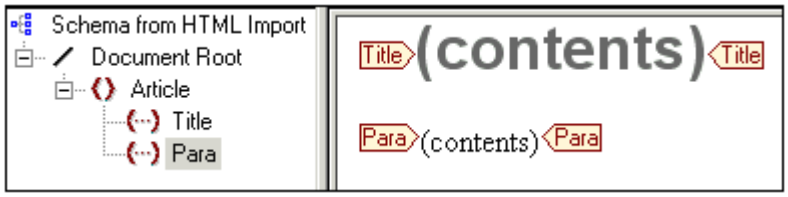
This is the main difference between (i) converting the selection to elements and (ii) surrounding the selection with an element node. But in the latter case, why is the `Title` element empty although the Design Document displays the text *A Sample Article* as the contents of the node? The text in the Design Document in this case is static text that will be included in the StyleVision Power Stylesheet. What you need to remember is that for HTML content to appear as element content in the XML document, the Design Document has to display that HTML content as a `(contents)` placeholder. All actual text in the Design Document—as is the case with StyleVision Power Stylesheets—is static text.

Undo's

Now undo your changes till the `Title` element is again represented by the `(contents)` placeholder in the Design Document. Notice that the Undo command works even though the file has been saved. **Do not delete** the `Title` element. If you delete the `Title` element that has been created to surround content, any static content that is contained in that element in the Design Document will also be deleted. In this case, this is not what we want.

Adding more children elements

Now convert the first HTML paragraph into a `Para` element that is a child of `Article`. Your schema tree and Design Document should look like this when you have finished:



Do not convert the second HTML paragraph into an element.

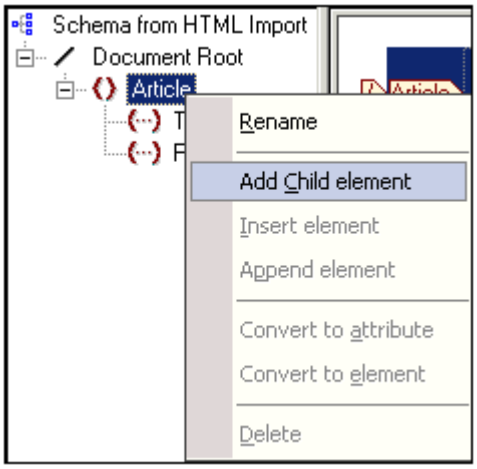
Additional elements

In the schema tree, you can also add elements that do not correspond to any HTML content. You can right-click a node and, from the context menu that pops up, choose whether to add a child element, insert a sibling element before the selected node, or append a sibling element.

Now do the following:

1. Right-click the `Article` node and, from the context menu, select **Add Child element**. Notice that Insert element and Append element are grayed out. This is because `Article` is the document element and cannot have any sibling.

 © 2005 Altova GmbH



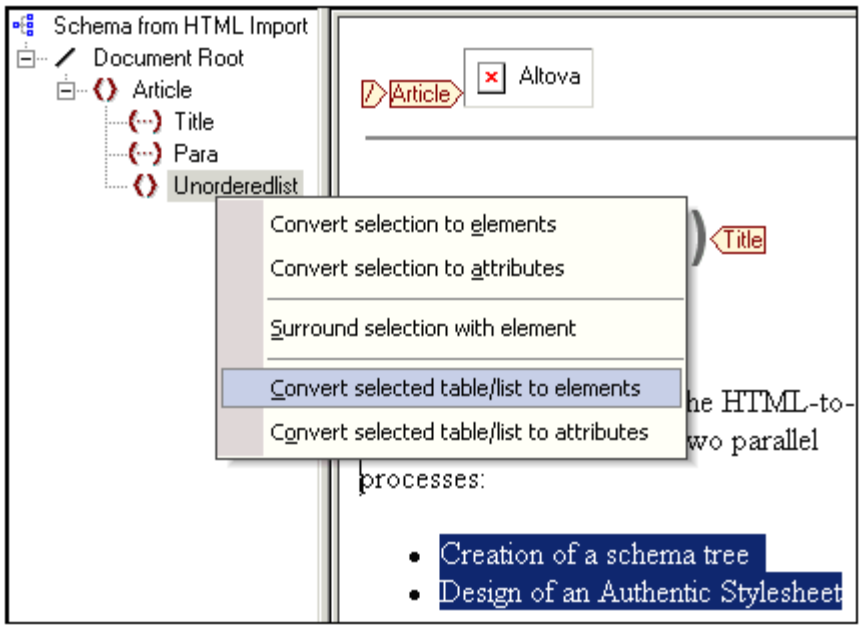
2. Name the newly created node `Unorderedlist`.

We will create the unordered list in the next section.

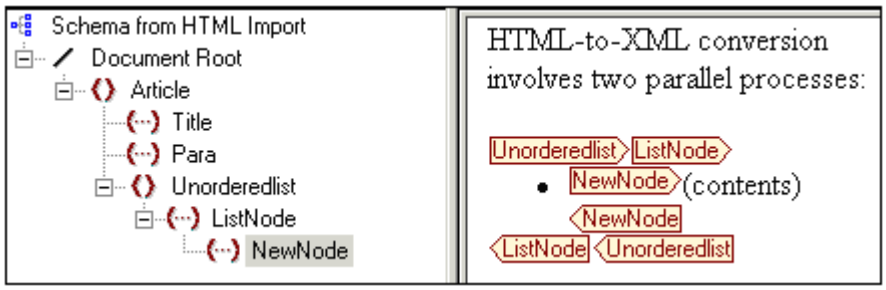
7.1.4 Importing an HTML list

You can import the unorderedlist HTML list as an XML structure in the following way:

1. Select the bulletted list in the Design Document and drop it into the schema tree as a child of `Unorderedlist`. The following menu pops up:



2. Select Convert selected table/list to elements, and in the next dialog, "as dynamic list". This creates a `ListNode` element that contains a child element `NewNode`.



   Notice how the list is displayed with a `(contents)` placeholder in the Design Document.
3. Rename the `NewNode` element to `Para`.

Now save your StyleVision document and look at the newly saved XML file.

```
<Article xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="C:\workarea\html2xml\article.xsd">
   <Title>A Sample Article</Title>
   <Para>This is a simple article, marked up in HTML. It is the example
         HTML document used to demonstrate how STYLEVISION can generate
         schema files, XML files, and stylesheets from an HTML document.
         It contains headlines, paragraphs, two lists, a table, an image,
         a hyperlink, horizontal rules, and inline italics. A number
         of these elements have attributes.</Para>
   <Unorderedlist>
      <ListNode>
         <Para>Creation of a schema tree </Para>
```

```
      </ListNode>
      <ListNode>
         <Para>Design of an Authentic Stylesheet </Para>
      </ListNode>
   </Unorderedlist>
</Article>
```

Note:

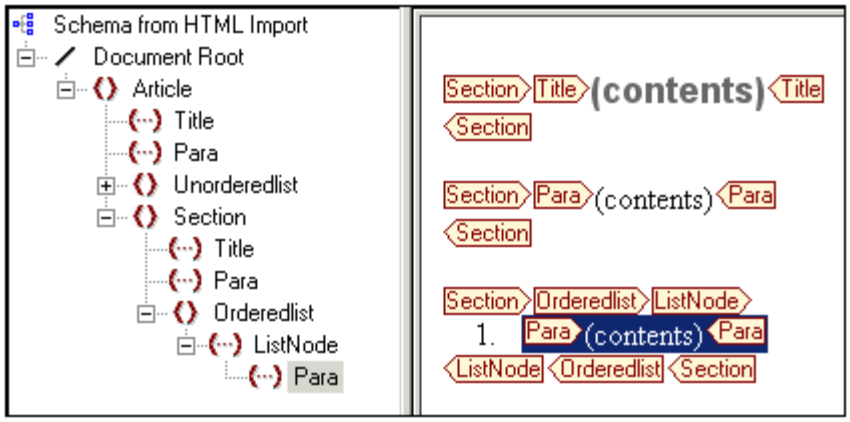
- Notice that for all elements created with `(contents)` placeholders, the corresponding HTML content is imported into the XML file. Notice also that the second HTML paragraph, for which no element was created, has not been imported into the XML document.
- In the schema file, `article.xsd`, notice that all the newly created element types are global elements. The `Para` element in `ListNode` is created by referencing the global element `Para`.
- If you need to use Global Templates in your StyleVision Power Stylesheet, then you should use the generated schema to build the StyleVision Power Stylesheet.

7.1.5 Creating a section

Now create a lower-level section for your article. To do this you repeat steps you have learned earlier.

1. Add an element called `Section` as a child of `Article`.
2. Select the HTML headline "Output" and create it as the `Title` child of `Section`.
3. Select the first paragraph after "Output" and create it as a `Para` child of `Section`.
4. Select the numbered list and import it as a list child of `//Section/Orderedlist`.

When you are done, your schema tree and Design Document should look like this:

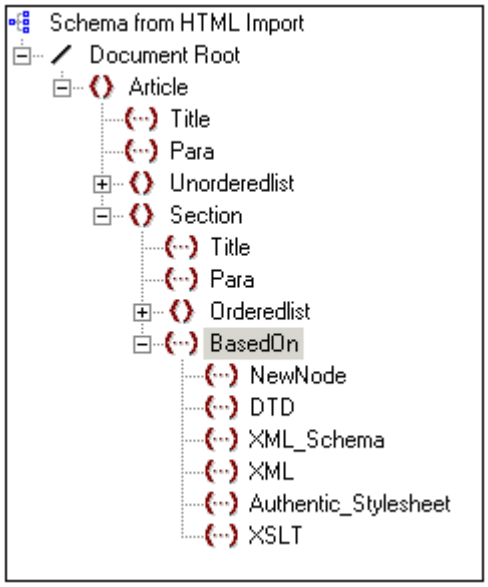


 © 2005 Altova GmbH

7.1.6 Importing HTML tables

You can import the HTML table into the XML document as follows:

1. Select any text fragment in the table and drag it into the Schema Window.
2. Drop it as a child of `Section`, and select Convert selected table/list to elements.
3. In the dialog that pops up select the dynamic table option; the top/down option; and the with-header option.
4. Rename `TableNode` to `BasedOn`. Your schema tree should now look like this:



Save the StyleVision document and view the table markup in the XML file:

```
<BasedOn>
   <NewNode>Schema Tree</NewNode>
   <DTD>•</DTD>
   <XML_Schema>•</XML_Schema>
   <XML>•</XML>
   <Authentic_Stylesheet> </Authentic_Stylesheet>
   <XSLT> </XSLT>
</BasedOn>
<BasedOn>
   <NewNode>Authentic Stylesheet</NewNode>
   <DTD> </DTD>
   <XML_Schema> </XML_Schema>
   <XML> </XML>
   <Authentic_Stylesheet>•</Authentic_Stylesheet>
   <XSLT>•</XSLT>
</BasedOn>
```

Let us make two changes to this markup: (i) Create a parent element called `Output` for the `BasedOn` element, and (ii) Convert the `NewNode` child element of `BasedOn` to a child attribute and rename the attribute to `Type`. This is how you would do it.

To create a parent for `BasedOn`, do the following in StyleVision:

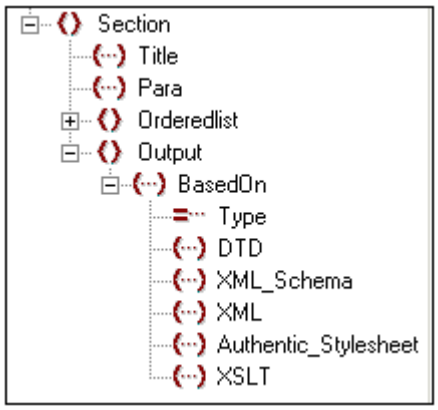
1. Right-click the `BasedOn` element in the schema tree.
2. Select Insert element.
3. Rename the new element to `Output`.
4. Drag `BasedOn` onto `Output` till the symbol indicating creation as child element appears, and then drop it.

Converting the `NewNode` child element to an attribute is described in the next section Creating schema attributes.

© 2005 Altova GmbH

7.1.7 Creating schema attributes

Converting elements to attributes
To convert the `NewNode` child element of `//Output/BasedOn` to a child attribute, right-click `NewNode` and select Convert to attribute. Then rename `NewNode` attribute to `Type`. This part of the schema tree will look like this:



The relevant XML file fragment will look like this:

```
<Output>
   <BasedOn Type="Schema Tree">
      <DTD>•</DTD>
      <XML_Schema>•</XML_Schema>
      <XML>•</XML>
      <Authentic_Stylesheet> </Authentic_Stylesheet>
      <XSLT> </XSLT>
   </BasedOn>
   <BasedOn Type="Authentic Stylesheet">
      <DTD> </DTD>
      <XML_Schema> </XML_Schema>
      <XML> </XML>
      <Authentic_Stylesheet>•</Authentic_Stylesheet>
      <XSLT>•</XSLT>
   </BasedOn>
</Output>
```

Creating HTML content as attributes
You can create attributes from HTML content by dragging the selected content to the element node for which you want to make this content an attribute. Create an attribute as follows:

1. Select the words "Altova GmbH" in the Copyright line at the bottom of the HTML document.
2. Drop it into the schema tree as a child of the `Article` element.
3. In the context menu, select **Convert selection to attributes**.
4. Name the attribute `Copyright`.

In the same way, select the text "2004" in the same Copyright line, and create it as an attribute `Year` of the `Article` element.

Moving nodes in the schema tree
Both these attributes will be appended to the bottom of the schema tree. If you like, you can drag them up the tree till they are just below `Article`.

7.1.8 Importing an HTML image

Now you will import an image into the XML file. StyleVision recognizes an HTML image and imports the value of the `<img>` element's `src` attribute as the content of element or attribute according to how you want this content created.

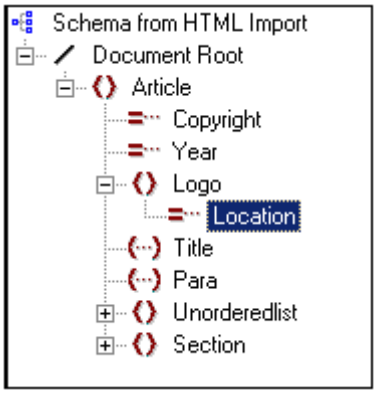
Creating the image source as attribute value
To import the image as an attribute value of an element, do the following:

1. Create an element `Logo` as a child of `Article`.
2. Select the image in the Design Document.
3. Drop it in the schema tree as a child of the `Article` element.
4. In the context menu that appears, select **Convert selection to attributes**.
5. Enter `Location` as the name of the attribute.

If you now check the XML file, you will find the element `Logo` at the bottom of the document, and that the value of the image's `src` attribute has been imported as the value of the `Location` attribute.

```
<Logo Location="altova.gif"/>
```

You can move the element `Logo` to alternative locations in the schema tree. Your schema tree should now look something like this:



Creating the image source as element value
Alternatively, you could have dragged the image into the schema tree and created it as a child element of any element. In this case the value of the image's `src` attribute is imported as the content of the element you create it as. For example:

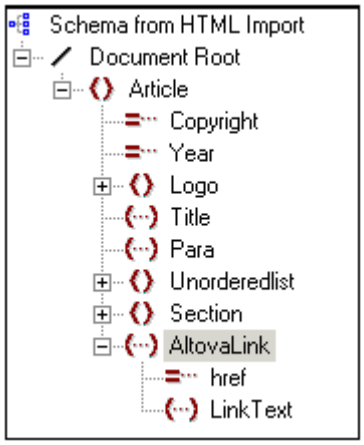
```
<LogoLocation>altova.gif</LogoLocation>
```

 © 2005 Altova GmbH

7.1.9 Converting HTML hyperlinks

StyleVision also allows you to import the URL and text of an HTML hyperlink. Do this in your document as follows:

1. Select the HTML hyperlink text "www.altova.com" in the Design Document.
2. Drag it into the schema tree and drop it as a child of `Article`.
3. Select **Convert selection to elements**.



4. Rename the nodes to the names shown in the screenshot.

The URL is imported to the `href` attribute of `AltovaLink` and the link text is imported as content of the element `LinkText`. The corresponding XML code should be:

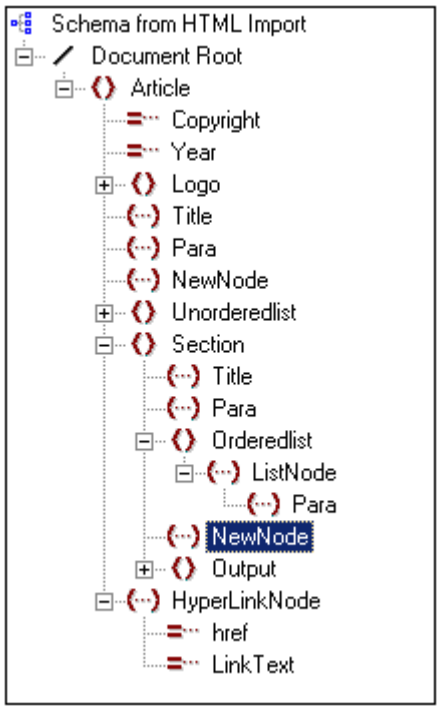
```
<AltovaLink href="www.altova.com">
   <LinkText>www.altova.com</LinkText>
</AltovaLink>
```

Note: If instead of selecting **Convert selection to elements** you had selected **Convert selection to attributes**, then the node now represented as the element `LinkText` would have been created as an attribute of `AltovaLink`.

7.1.10 Completing the import

Same-name siblings
If you look at the Design Document, you will notice that there are two HTML paragraphs that have not been imported. In order to import these into the XML document, do the following:

1. Convert each HTML paragraph into an element at the correct location in the schema tree giving each a different name. In the screenshot below, you will see that both the paragraphs have been imported as `NewNode` elements, in one case as `/Article/NewNode` and in the other as `//Section/NewNode`.



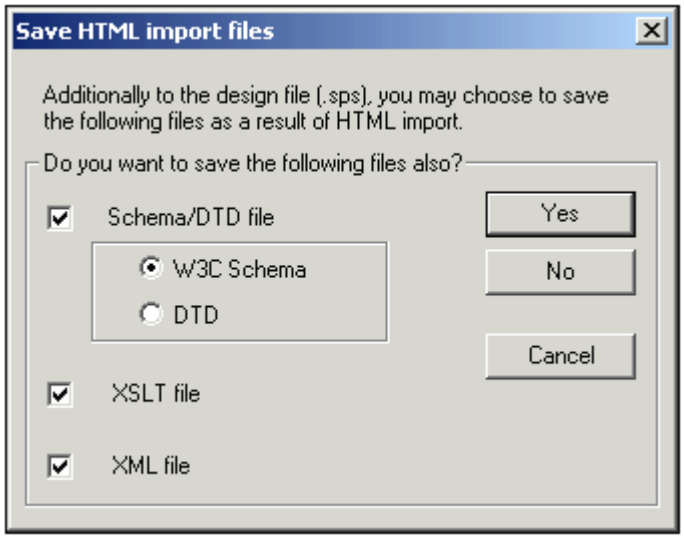
2. Save the StyleVision document.
3. Open the XML document and change the element names as required, for example, to `Para`.

Note: In this case, you should modify the generated schema to remove `NewNode` and possibly modify the properties of the `Para` element.

© 2005 Altova GmbH

7.1.11 Generating the output

After you have completed the schema tree, you can generate and save the schema, XML file, StyleVision Power Stylesheet, and XSLT file. To do this, click **File | Save Design**. The following dialog appears:


Save HTML import files
Additionally to the design file (.sps), you may choose to save the following files as a result of HTML import.
Do you want to save the following files also?
Schema/DTD file
W3C Schema
DTD
XSLT file
XML file
Yes
No
Cancel

Select the file types you wish to save. The schema file and StyleVision Power Stylesheet must be saved. Take a look at the generated XML file. If you wish to try out some changes, go back to StyleVision and make the desired modifications, and re-save.

Note: In order to make modifications and re-save the StyleVision Power Stylesheet, you **must not** close the StyleVision Power Stylesheet, that is, the StyleVision Power Stylesheet must still be open in StyleVision.

7.2 HTML-to-XML Quick Reference

This Quick Reference section covers all the commands you need to use the HTML-to-XML conversion utility of StyleVision effectively. The required HTML-to-XML-specific commands are available in the **File** and **HTML Import** menus.

In the **File** menu, the **Save Design** command, in addition to saving the StyleVision Power Stylesheet (SPS) file, also prompts you about whether you wish to save the SPS, XSLT and/or XML files.

Complementary information about the HTML-to-XML utility can be found in the HTML-to-XML Tutorial and in the User Reference section of the Altova StyleVision 2005 manual.

© 2005 Altova GmbH

7.2.1 Importing the HTML file

Importing the HTML file
To import an HTML file into StyleVision for conversion to XML, click the menu item **HTML Import | Open HTML file...** or **File | Open HTML file...**, and select the required HTML file. The selected file will be imported into the Main Window, and a schema tree with a document element called `Root` will be created in the Schema Window.

7.2.2 Building the schema tree: 1

Creating HTML content as child or sibling
When you select HTML content and drag it to a node in the schema tree, you could create it as a child element or attribute, or as a sibling element or attribute (a sibling attribute is an attribute of the selected node's parent). The icons that appear in either case are as follows:

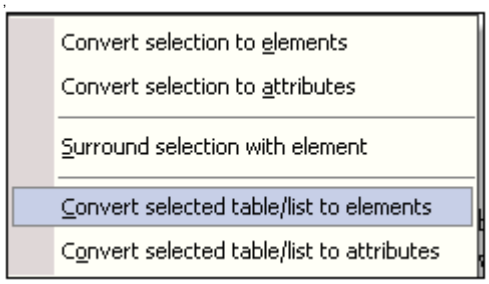

Indicates creation as child node of the highlighted schema tree node.


Indicates creation as sibling node of the highlighted schema tree node.

Some nodes can have both children and siblings. In these cases, the icon for accepting siblings appears when the cursor is moved to the left side of the node. To create a child or sibling node, release the mouse button when the appropriate icon appears.

Converting HTML content to an XML node type
After selecting whether the HTML content is to be created as a child or a sibling, you select whether it is to be created as an element with or without the HTML content; or as an attribute. If the HTML content is a list or table, you have the additional options of whether the list or table should be converted to a corresponding element or attribute structure.

After you drag HTML content to the schema tree, the following menu pops up when you release the mouse button. This menu can also be accessed by selecting the following StyleVision menu options **HTML Import | Add to parent**, **HTML Import | Insert**, and **HTML Import | Append**.


Convert selection to elements
Convert selection to attributes
Surround selection with element
Convert selected table/list to elements
Convert selected table/list to attributes

The commands of this context menu are described below.

Convert selection to elements
The HTML content is inserted as an element or elements into the tree. How many elements are created depends on the structure of the object. The HTML content is created as a `(contents)` placeholder and becomes the content of the element node in the generated XML document.

Convert selection to attributes
The HTML content is inserted as an attribute or as attributes into the tree. The HTML content is created as a `(contents)` placeholder and becomes the content of the attribute node in the generated XML document.

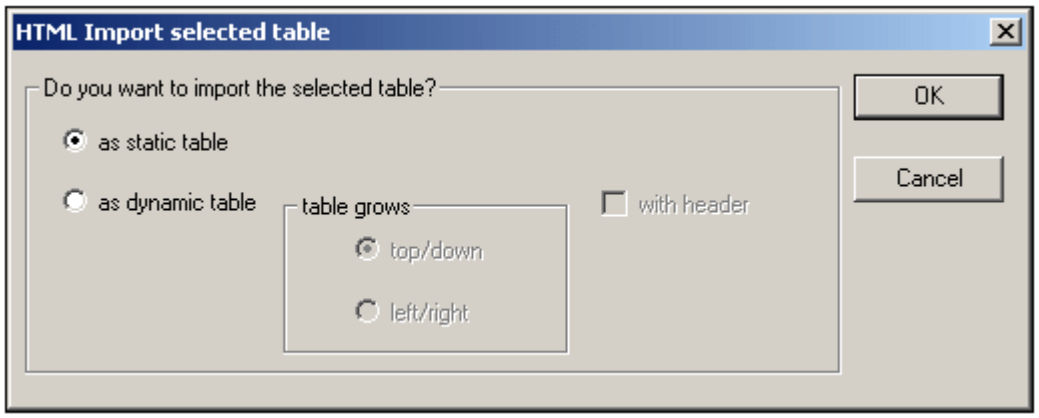
Surround selection with element
The HTML content is surrounded by an element. No `(contents)` placeholder is created, and the HTML content does not appear in the generated XML document. The content within the newly created node is static and is part of the Root Template of the generated StyleVision Power Stylesheet and XSLT. You can format it as you do any template in StyleVision.

Convert selected table/list to elements
The whole HTML table or list will be imported as elements into the XML tree.

 © 2005 Altova GmbH

Convert selected table/list to attributes
The whole HTML table or list will be imported as attributes into the XML tree.

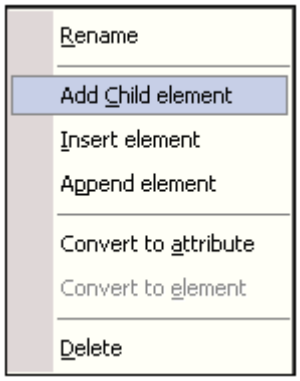
When the selected HTML content is a table and you select either **Convert selected table/list to elements** or **Convert selected table/list to attributes**, the following dialog box appears:



Creating the node as part of a table element is useful if the object is, say, a table cell and you wish to create an XML structure corresponding to the HTML table structure. For a description of static and dynamic tables, see Tables in StyleVision.

7.2.3 Building the schema tree: 2

You do not have to drag HTML content into the schema tree in order to be able to create elements. You can directly create elements as children or siblings of a selected node, as well as convert a selected node to an element or attribute. These commands are available in a context menu that pops up when you right-click a node.



The commands of this menu are explained below.

Rename
Allows you to rename the selected node.

Add child element
Adds a child element to the selected node.

Insert element
Inserts a sibling element before the selected node.

Append element
Appends a sibling element after the selected node.

Convert to attribute
Converts the selected element to an attribute.

Convert to element
Converts the selected attribute to an element.

Delete
Deletes the node in the schema tree. If the node was created from HTML content and if the HTML content has been created as a `(contents)` placeholder (by selecting Convert selection to elements or Convert selection to attributes when the HTML content was dropped into the schema tree), then the node is deleted in the Design Document and the original HTML content is restored. Otherwise the node is deleted from the Design Document but the original HTML content is not restored.

 © 2005 Altova GmbH

7.2.4 Points to note

The following important points should be noted:

- If you wish to edit either the schema, XML file, XSLT stylesheet, or StyleVision Power Stylesheet after having saved them, you must open the StyleVision Power Stylesheet to edit and then re-save the required files. The HTML content available for editing at this stage is what is available in the StyleVision Power Stylesheet as static text.
- Note that unlimited Undo is supported, so you can go back as many steps as you like. You can also, therefore, save the schema, XML file, and stylesheets as often as you like, without having to worry about not being able to undo your changes.
- You can only create a Root Template for the XML document. However, all the elements you create in the schema tree are declared as global elements (or as references to global elements). If you need to use global templates for your StyleVision Power Stylesheet, you should save the StyleVision Power Stylesheet and then re-open it: the global elements will now be accessible in the Global Templates list.
- Attributes are not allowed to be added to the schema tree by right-clicking a node. If you need to add an attribute independently of HTML content, right-click the relevant node, create an element, and then convert it to an attribute (by right-clicking and selecting Convert to attribute).

Chapter 8

User Reference

8 User Reference

This section contains a complete description of StyleVision windows and menu commands. It is divided into the following broad parts:

- A description of symbols, icons, Text and Block Style Windows, and options for Formatting.
- All menu commands
- Context menus

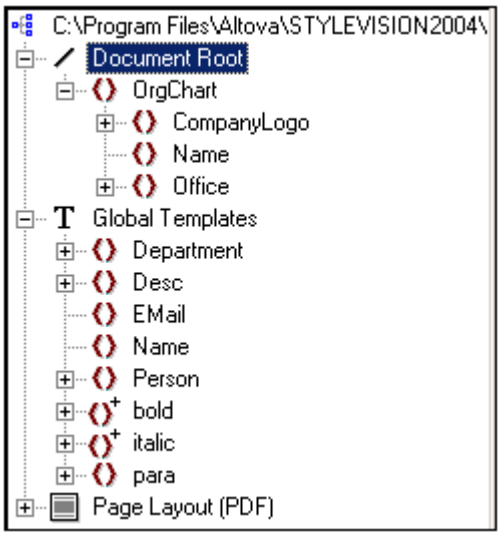
While the User Reference section contains a description of individual commands, the mechanisms behind various StyleVision features are explained in detail in the How To Use section. For complex issues, refer also to the Tips and Troubleshooting sections.

© 2005 Altova GmbH

8.1 Symbols and icons

The following symbols and icons are used in StyleVision.

Schema Window symbols and icons
The Schema Window contains three parts, each of which is displayed as a tree: the Schema; Global Templates; and Page Layout (PDF). The Page Layout (PDF) tree is available only in the Enterprise Edition.

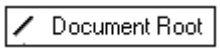


When you click on a component entry in any of the three trees, that component entry is highlighted and the Design Window displays and highlights that tree or component.

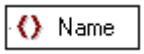
Schema tree



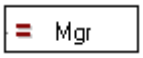
Schema name and folder.



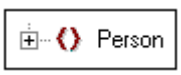
The document root (which is not the same as the document element, and occurs above it).



Element.

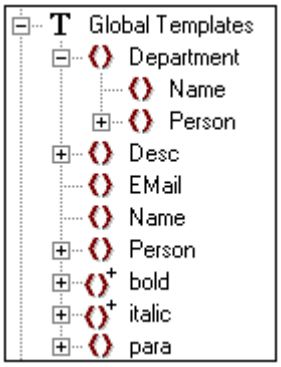


Attribute.



Element with child elements. Double-clicking the element or the +/- symbol to its left causes the element to expand/collapse.

Global templates
The entries in the global templates list are the global elements in an XML Schema or all elements in a DTD.

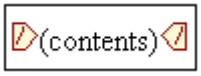


Clicking an entry creates a default global template in the StyleVision Power Stylesheet. A + sign appears on the right of the element symbol 

indicating that a global template for this element now exists. The default template (which merely processes the element's contents (with an `xsl:apply-templates` rule)) is now displayed in the Design Window, where it can be modified. If the global template for an element is removed (by right-clicking the global template entry and pressing Remove Global Template), the + sign disappears.

Design Window symbols and icons

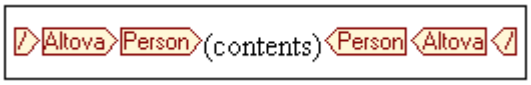
Element and attribute symbols can be moved around using **drag and drop** in the Design window. The standard Cut, Copy, and Paste commands can also be used. The following symbol and icons are displayed in the Design Window



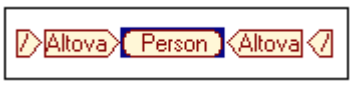
Root element tags.

`(contents)`

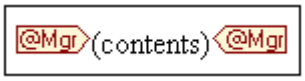
Placeholder for XML content of that element. It corresponds to the `xsl:apply-templates` rule. You can select the placeholder and apply formatting to it; this formatting will be applied to the XML content of the element. You can also delete the `(contents)` placeholder.



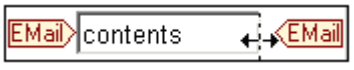
Start and end tags of elements. An element inserted from the schema tree is inserted with the correct ancestors relative to the location into which it is dropped.



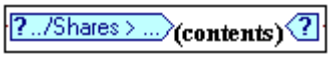
A contracted element tag. Tags can be expanded/contracted by double-clicking.



Attribute tags contain the `@` prefix.



An input field. The width can be adjusted by dragging the right border.



Conditional template tags are blue and contain a question

 © 2005 Altova GmbH

mark.



An Auto-Calculation is indicated by an equals-to sign followed by the first characters of the XPath expression used for the Auto-Calculation.

Toolbar items

Most toolbar items are intuitive and display a tooltip when you mouseover. Some are specific to StyleVision.



Select XSLT 1.0. Uses XSLT 1.0 as the XSLT language version of the StyleVision Power Stylesheet (and any generated XSLT stylesheet). Also selects the XSLT 1.0 processor for internal processing of StyleVision Power Stylesheet or XSLT stylesheets. Enabled only in Design View.



Select XSLT 2.0. Uses XSLT 2.0 as the XSLT language version of the StyleVision Power Stylesheet (and any generated XSLT stylesheet). Also selects the XSLT 2.0 processor for internal processing of StyleVision Power Stylesheet or XSLT stylesheets. Enabled only in Design View.



Authentic Node Settings. Brings up the Authentic Node Settings dialog for the selected node.



The Pre-Defined Format combo box. Selecting a format from this drop-down box applies it to the selected element. All content, when initially created, is assigned the Normal format by default.



Insert Static SPS table. See Tables for more details on the various types of tables available in StyleVision.



Edit Stylesheet Parameters. Brings up the Edit Parameters dialog, in which you can declare global parameters for the stylesheet.



Auto-add Date-picker. If this command is toggled on, the date-picker will be inserted when a node of datatype `xs:date` is dropped into the Design Document.



Define XML Entities. Enabled in Authentic Editor View only. Pops up the Define Entities dialog, which enables you to define entities you can subsequently insert into the XML document by right-clicking and selecting the required entity.



Validate XML (F8). Enabled in Authentic Editor View only. Validates the XML document.



Save Authentic XML Data. Saves modifications you make to the Working XML File in Authentic Editor View.

Also see: StyleVision interface

© 2005 Altova GmbH

8.2 Formatting

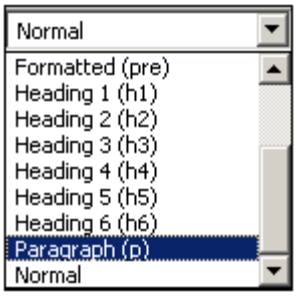
Formatting can be applied to a selected component in one of, or any combination of, the following three ways:

- By selecting a pre-defined format
- By specifying values for Block Style properties
- By specifying values for Text Style properties

To assign formatting to a component, select it, and then use one or more of the three methods listed. Each method is explained below. The properties in the Block Style Window and Text Style Window are broadly based on HTML styling principles.

Pre-defined formats
StyleVision has a number of pre-defined formats, the properties of which you cannot change. When you apply a pre-defined format to a component, the formatting markup is created **around the selected component**. When an element or attribute is created in the Design Document, it is assigned the Normal format by default. Any pre-defined format must be explicitly assigned.

Pre-defined formats can be assigned by clicking **Insert | Format**, and then the required format, or by selecting the required format from the Format drop-down list in the Toolbar (shown below).



Note: Assigning a pre-defined format to a component makes that component a block component.

The pre-defined formats used in StyleVision have either one or both of the following two components:

- a text-formatting component
- a spacing component.

For example, the pre-defined `para (p)` format has a spacing component only; it puts vertical space before and after the selected component, and does not apply any text formatting. On the other hand, the pre-defined `Heading 1 (h1)` format has both a text-formatting component and a spacing component.

The following points about pre-defined formats should be noted:

- The spacing component of a pre-defined format applies for any type of StyleVision Power Stylesheet component, but the text formatting only if it can be applied. For example, if you select an image, and apply a pre-defined format of `Heading 1 (h1)` to it, then the spacing component will take effect, but the text-formatting component will not.
- The text-formatting component of pre-defined formats does not apply to data-entry devices.
- Only one pre-defined format applies to a component at any given time.

Block Style properties
Block style properties are assigned in the Block Style window. When an element/attribute is created, it is created with the default pre-defined format of `Normal` and is not a block. To make the element/attribute a block, assign it any pre-defined format other than `Normal`.

Caution: When applying block styles within a table cell, remember that the table cell is a block. It may be that the component in the table cell is also a block (you can find out by checking its pre-defined formatting). If the component is also a block, be sure to select the correct block (i.e. table-cell block or component block). Note that you can also apply table formatting through the Table Properties dialog.

Tip: You do not need to make a component a block in order to apply certain block styles (such as spacing) to it. You can apply such block styles in the Text Style window.

Text Style properties
Text style properties are assigned in the Text Style window, and the properties are assigned as **inline** markup surrounding the component in the markup of the output. Text style properties, therefore, override block style properties and pre-defined formats if there is a conflict.

Formatting conflicts
If there is a formatting conflict, i.e. a formatting property is defined using more than one of the above methods, the conflict is resolved using the same principle used in Cascading Style Sheets: inline markup has priority over higher-level markup. The priority for the different methods is as follows:

1. Text Style (since this is inline markup)
2. Block Style (since this applied to the block)
3. Pre-defined format (the formatting is inherent in the element name, e.g. `h2` and `div`)

The Return key and pre-defined formats
In Authentic View, when the Return key is pressed within the contents of an element having a pre-defined format, the current element instance and its block are terminated, and a new element instance and block are inserted at that point. This property is useful, for example, if you want the Authentic View user to be able to create a new element, say a paragraph-type element, by pressing the Return key. To obtain this behavior in Authentic View, you would have to assign the `(contents)` placeholder a pre-defined format of `Paragraph(p)`.

Removing pre-defined formats
To remove a pre-defined format, select the component with the pre-defined format you want to delete, and change the pre-defined format to `Normal`. If the pre-defined format has been inserted at a non-node location, then when you place the cursor at this location the selected pre-defined format will be shown in the Pre-defined Format combo box. Change this option to `Normal`.

Also see: Document design process, Text and Block Style Windows, and Input Formatting.

 © 2005 Altova GmbH

8.3 File

The **File** menu contains commands for working with StyleVision Power Stylesheets and related files. Some of these commands, such as **Open** and **Save** will be familiar from other Windows software products.

8.3.1 New...

 **Ctrl+N**

The **New...** command allows you to create a new StyleVision Power Stylesheet or SPS document from a **schema** (i.e. an **XML Schema** or **DTD**). The schema is loaded into the Schema Window in a tree form. The **Document Element** or Root Element is created immediately below the **Document Root**. After the schema is loaded, you can start designing the StyleVision Power Stylesheet by dragging and dropping elements from the Schema Window into the Design Window.

Creating the StyleVision Power Stylesheet from an XML Schema

- If there is more than one global element in the XML Schema, then the Document Root will be empty. This is because it is not known (to the application) which one of the global elements should be used as the Document Element. To indicate this to the application, assign a Working XML File (**File | Assign Working XML File**), which is based on the same XML Schema, to the StyleVision Power Stylesheet. As soon as you do this, the Document Element of the XML file is selected as the Document Element of the StyleVision Power Stylesheet, and the schema tree is displayed in the Schema Window. If you do not have a Working XML File handy, create a rudimentary XML document with the desired Document Element.
- Only elements defined as global elements in the XML Schema are created in the Global Templates list in the Schema Window.

Creating the StyleVision Power Stylesheet from a DTD

- A valid DTD must define a single Document Element. So, if you create a StyleVision Power Stylesheet from a DTD, the Document Element is known and a schema tree is displayed without you having to assign a Working XML File.
- All elements defined in the DTD are created in the Global Templates list in the Schema Window.

Note:

- If your schema is not well-formed or is invalid, you may get an error message when you try to load it. In this case, open the schema and try to resolve the problem in it.
- Only one file can be open at a time in StyleVision. A newly opened file replaces the previously opened file. If you have not saved changes to the previously opened file you will be prompted about whether you wish to save your changes or not.

 © 2005 Altova GmbH

8.3.2 Open...

 **Ctrl+O**

The **Open...** command allows you to open a StyleVision Power Stylesheet (SPS file). The familiar Open dialog of Windows systems is opened and allows you to select a `.sps` file type.

If a Working XML File was assigned to the StyleVision Power Stylesheet during a previous session, then the Authentic Editor View will be also be enabled. Otherwise you will have to assign a Working XML File in order to enable Authentic Editor View.

Note

- Only one file can be open at a time in StyleVision. Opening a new file causes the currently open file to close. If you have not saved changes to the currently open file, you will be prompted about whether you wish to save your changes or not. If the new file cannot be opened for whatever reason, the previously open file will have been closed and the interface will be empty.

8.3.3 Close

The **Close** command closes the currently open StyleVision Power Stylesheet file. Note that only one file can be open at a time in StyleVision.

© 2005 Altova GmbH

8.3.4 Import HTML File...

The Import HTML File... command allows you to import an HTML file into StyleVision. You can then convert this HTML file to an XML file in StyleVision. How one does this is explained in the HTML to XML conversion section of this documentation.

8.3.5 Assign Working XML File...

The **Assign Working XML File** command assigns an existing XML file to the currently open StyleVision Power Stylesheet (SPS file). The name of the Working XML File (including its path) is displayed in the title bar of StyleVision.

The Working XML File must conform to the same schema that is used for the StyleVision Power Stylesheet. It is used to provide the Authentic Editor View and HTML preview of the output you get using the generated XSLT stylesheets (see XSLT and StyleVision Power Stylesheets and Uses of StyleVision Power Stylesheets). Assigning a Working XML File is also the only way to resolve ambiguity about which global element in an XML Schema StyleVision should select as the Document Element (see File | New...). You will not be able to see an Authentic Editor View or HTML Preview without assigning a Working XML File. If a Working XML File has not been assigned, and you click any preview tab, you will be prompted to select a Working XML File.

Changing the Working XML File
To change the Working XML File, click **File | Assign Working XML File**, select the new Working XML File. The new Working XML File is assigned. Save the StyleVision Power Stylesheet if you want to keep the change.

Unassigning the Working XML File
Open the StyleVision Power Stylesheet in an XML editor, such as XMLSpy 2005, or a plain text editor, such as Notepad. The root element of the StyleVision Power Stylesheet is `<structure>`. Delete the contents of the `workingxmlfile` attribute of `structure`, so that the attribute has an empty value. The start tag of the `structure` element should now look something like this:

```
<structure version="2" schemafile="test.xsd" workingxmlfile=""
templatexmlfile="">
```

Save the StyleVision Power Stylesheet. The Working XML File is now unassigned. Note that the Authentic Editor View in StyleVision cannot be used if no Working XML File is assigned.

 © 2005 Altova GmbH

8.3.6 Edit Schema Definition in XMLSpy

This command opens the schema file in XMLSpy 2005 so that you can edit it. After you have edited and saved the modifications to your schema, you must reload the StyleVision Power Stylesheet so that the schema in the StyleVision Power Stylesheet is updated with the changes you made.

8.3.7 Encoding...

The **Encoding...** command pops up a dialog in which you can select the encoding of your output documents. Changing the encoding in this dialog changes the encoding for the currently active SPS. The default encoding (for new StyleVision Power Stylesheets) is specified in the Encoding tab of the Options dialog.

© 2005 Altova GmbH

8.3.8 Save Design

 **Ctrl+S**

The **Save Design** command saves the currently open StyleVision file as a StyleVision Power Stylesheet (with the file extension **.sps**).

Note: When you create an XML Schema from an imported HTML file, a StyleVision Power Stylesheet is simultaneously created. To save additional files created or generated from the imported HTML file, click the Save Design command. This pops up a dialog that gives you the option of saving the created XML Schema, generated XSLT stylesheets, and generated XML file. to save th

8.3.9 Save Design As...

The **Save Design As...** command shows the familiar Save as... dialog of Windows systems and prompts you for the name of the StyleVision Power Stylesheet to be saved and the location where you want it saved. The newly saved file becomes the current file in StyleVision.

© 2005 Altova GmbH

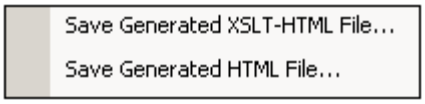
8.3.10 Save Authentic XML Data...



In Authentic Editor View, you can edit the Working XML File related to the StyleVision Power Stylesheet. The **Save Authentic XML Data...** command saves these modifications to the Working XML File . Alternatively, you can edit an XML document in the Authentic View of Altova XMLSpy 2005 or Altova Authentic 2005 Desktop Edition.

8.3.11 Save Generated Files

The **Save Generated Files** command pops up a submenu which contains options for saving the following files. For the relationship between the StyleVision Power Stylesheet and the various generated files, see Uses of StyleVision Power Stylesheets and StyleVision and output.



Save Generated XSLT-HTML File...
The Save Generated XSLT-HTML File... command generates an XSLT file for HTML output from your StyleVision Power Stylesheet. You can use this XSLT file subsequently to transform an XML document to HTML.

Save Generated HTML File...
The Save Generated HTML File... command generates an HTML file. This operation requires two input files:

- The Working XML File assigned to the currently open SPS file. If no Working XML File has been assigned, the Save Generated HTML File... command is disabled.
- An XSLT file, which is automatically generated from the currently open SPS file.

© 2005 Altova GmbH

8.3.12 Print Preview...



The **Print Preview...** command is enabled only in Authentic Editor View. It opens a window containing a preview of the Authentic View of the Working XML File.

8.3.13 Print...

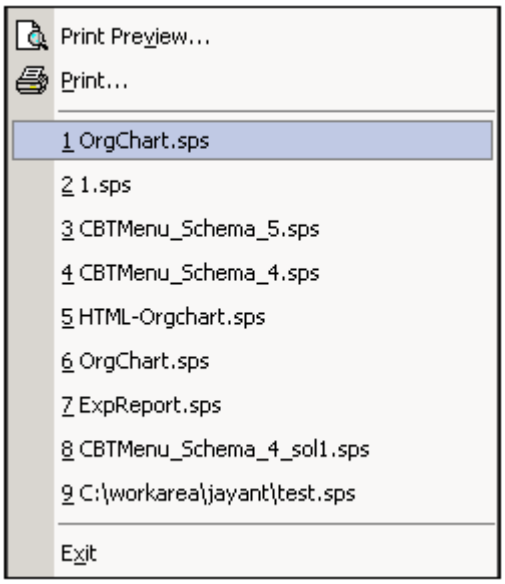


The **Print...** command prints the Authentic View of the Working XML File.

© 2005 Altova GmbH

8.3.14 Most recently used files

The list of most recently used files, shows the file name and path information for the nine most recently used files, which you can select with the mouse.



To access these files using the **keyboard**, press **ALT+F** to open the File menu, and then the number of the file you wish to open; for example, pressing **1** will open the first file in the list, **2** the second file, and so on.

8.3.15 Exit

The **Exit** command is used to quit StyleVision. If you have an open file with unsaved changes, you will be prompted to save these changes.

© 2005 Altova GmbH

8.4 Edit

The **Edit** menu contains the **Undo** and **Redo** commands which allow you to undo or restore your previous actions. Other editing commands include **Find, Find Next,** and commands to edit stylesheet parameters.

The **Edit** menu also contains other standard editing commands such as **Cut** (Ctrl+Del), **Copy** (Ctrl+C), **Paste** (Ctrl+V), and **Delete** (Del). These are not described in this section.

Commands that are not applicable at a given location, or for a given selection, are grayed out in the menu. The relevant commands are also available via the context menu that appears when you right-click a component or right-click at a cursor insertion point inside the document. Additionally, some commands are available as shortcuts and/or toolbar icons.

8.4.1 Undo

 **CTRL+Z**

The **Undo** command enables you to undo an editing change and contains support for an unlimited number of Undos. Every action can be undone and it is possible to undo one command after another till the first action that was made since the document was opened.

Also see: Redo.

© 2005 Altova GmbH

8.4.2 Redo

 **CTRL+Y**

The **Redo** command allows you to redo any number of previously undone commands. By using the Undo and Redo commands, you can step backward and forward through the history of commands.

Also see: Undo.

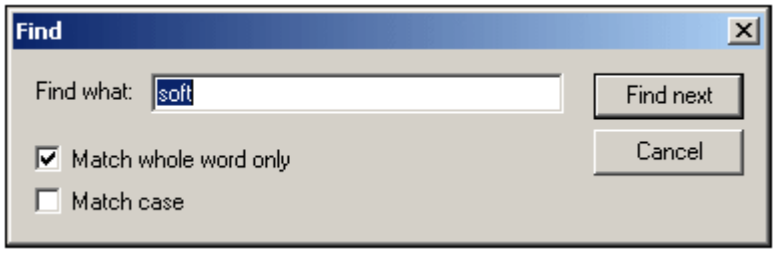
8.4.3 Find



Ctrl+F

The **Find** command allows you to find words or fragments of words in the Authentic Editor View of the Working XML File, and in the XSLT-for-HTML stylesheet. The **Find** command is enabled only when the Authentic Editor View or XSLT stylesheet tabs are selected.

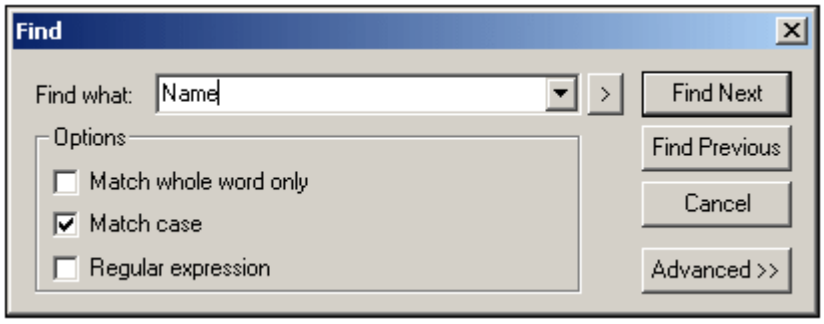
Authentic Editor View
Clicking the **Find** command when the Authentic Editor View tab is selected pops up the following dialog:



Note the following:

- In the Authentic Editor View, only the XML data is searched. Text that is inserted via the XSLT is not searched.
- To match the entry with whole words in the XML document, check "Match whole word only". For example, an entry of `soft` will find only the whole word `soft`; it will not find, for example, the `soft` in `software`.
- To match the entry with fragments of words, leave the "Match whole word only" check box unchecked. Doing this would enable you, for example, to enter `soft` and `software`.
- To make the search case-insensitive, leave the "Match case" checkbox unchecked. This would enable you to find, say, `Soft` with an entry of `soft`.

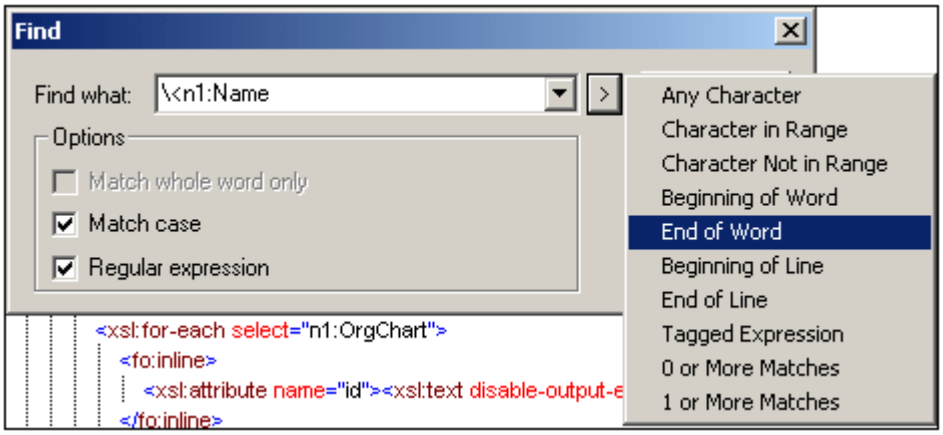
XSLT-for-HTML stylesheet
When the XSLT-for-HTML tab is selected, clicking **Find** pops up the following dialog:



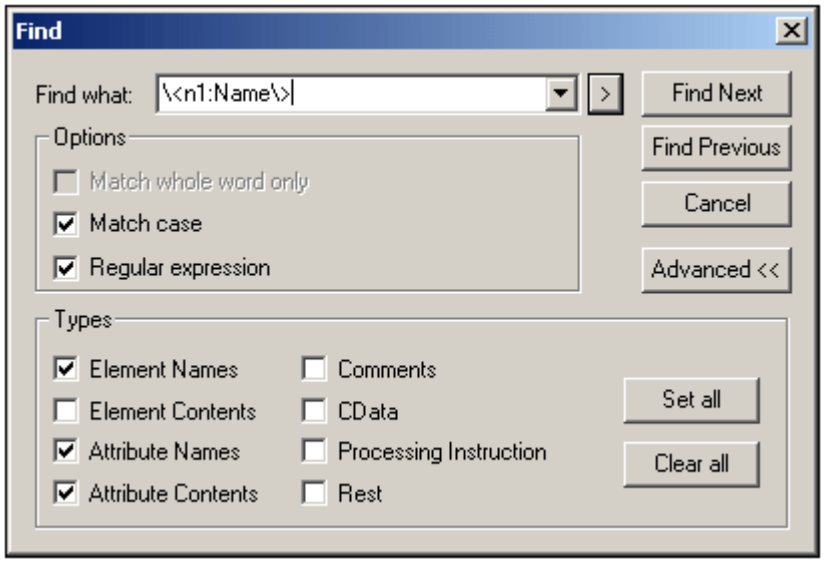
The following points should be noted:

- In the XSLT-for-HTML stylesheet, all text is searched.
- To enter a regular expression as the search term, check the Regular expression check box. You can create a regular expression with the help of a menu that pops out when you click the right-pointing arrowhead near the search term entry field.

© 2005 Altova GmbH



- To set restrictions on what part of the document to search, click the Advanced button. More options become available in the dialog (shown below):



Select the types of document content you wish to search by checking the appropriate check box.

8.4.4 Find Next



F3

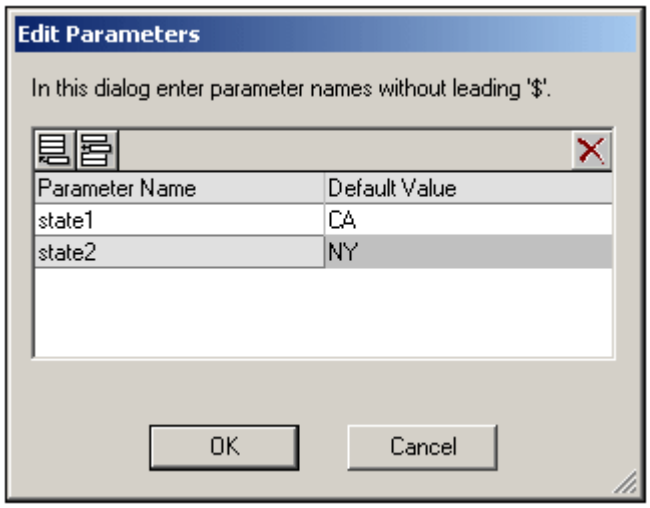
The **Find Next** command repeats the last Find command to search for the next occurrence of the requested text. See Find for a description of how to use the search function.

© 2005 Altova GmbH

8.4.5 Edit Stylesheet Parameters...



The **Edit Stylesheet Parameters...** command enables you to declare and edit parameters and their default values. The command is available in both the Design Document view and the Authentic Editor View. When you click this command, the Edit Parameters dialog (shown below) pops up.



The following points should be noted:

- You can insert, append, edit and delete parameters for the entire stylesheet.
- Parameter names must begin with a letter, and can contain the characters `A` to `Z`, `a` to `z`, `0` to `9`, and the underscore.

Also see:

- For an overview of how parameters are to be used, Parameters in the How To Use section.

8.4.6 Select All

The **Select all** command selects the entire contents of the Design Document window.

© 2005 Altova GmbH

8.5 Insert

The **Insert** menu provides commands enabling you to insert various items into the Design Document. The relevant commands are also accessible via the context menu, which appears when you place the cursor or select a component in the Design Document. In some cases, a command is additionally available as a shortcut and/or as an icon in the toolbar.

Note: The Insert commands are used to build the StyleVision Power Stylesheet and are therefore available in the Design Document view only.

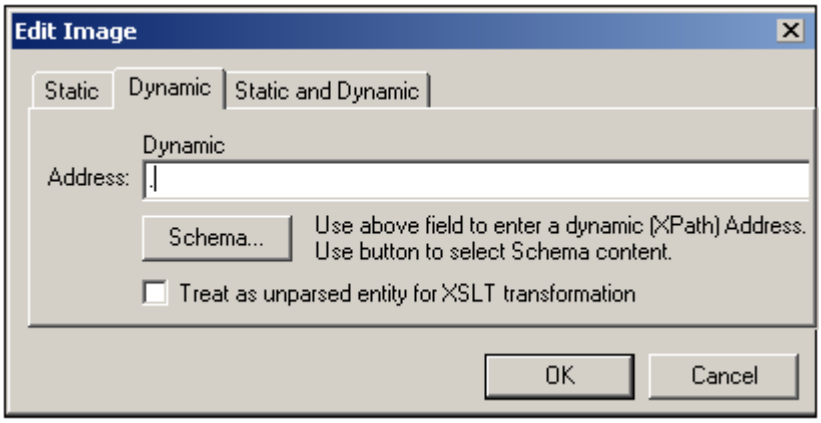
8.5.1 Image...



The **Image...** command allows you to insert an image using an image location address that either comes from the XML document (dynamic) or is entered by you directly in the StyleVision Power Stylesheet (static).

To insert an image, do the following:

1. Click **Insert | Image...** or the Insert Image toolbar icon. The Insert Image dialog (shown below) appears.



2. Select the required tab (Static, Dynamic, or Static and Dynamic), and enter the address of the image location and/or the XPath expression that locates the image address in the XML document. The screenshot above shows how a Static and Dynamic address is entered.

Image address in XML documents
If the image address (a URL), or a part of the address, is contained in a node in the XML document, this address can be used by selecting the node that contains it. You do this by entering an XPath expression that selects the required node in the Dynamic tab (or Static and Dynamic tab) of the Insert Image dialog. Note that the URL contained in the XML node can be an absolute or relative URL. If relative, it must be relative to the location of the FO file. When you are working with StyleVision, a temporary FO file is created in the same folder as the SPS file, and it is this file that is used to generate the PDF Preview. So the image address in the XML file should be relative to the location of the SPS file if you wish to see the image in PDF Preview. If you wish to generate and save the PDF, make sure that the image file referenced in the FO document is correctly referenced.

Using unparsed entities
If you are using a DTD as the basis of your StyleVision Power Stylesheet, then, for the dynamic part of an image address, you can use the URI declared for an unparsed entity in the DTD. For details of how to use unparsed entities, see Using unparsed entity URIs.

Also see: Including an image in the tutorial.

 © 2005 Altova GmbH

8.5.2 Paragraph



The **Paragraph** command inserts an HTML paragraph <p> element around the selected component. A component is considered selected for this purpose when the entire node is selected (by clicking either of its tags) or when the cursor is placed inside the component.

Also see: Formatting the components.

8.5.3 Format

The **Format** command allows you to assign a predefined format to the selected node. The available predefined formats can also be selected from the combo box in the toolbar.



Also see: Formatting and Formatting the components.

© 2005 Altova GmbH

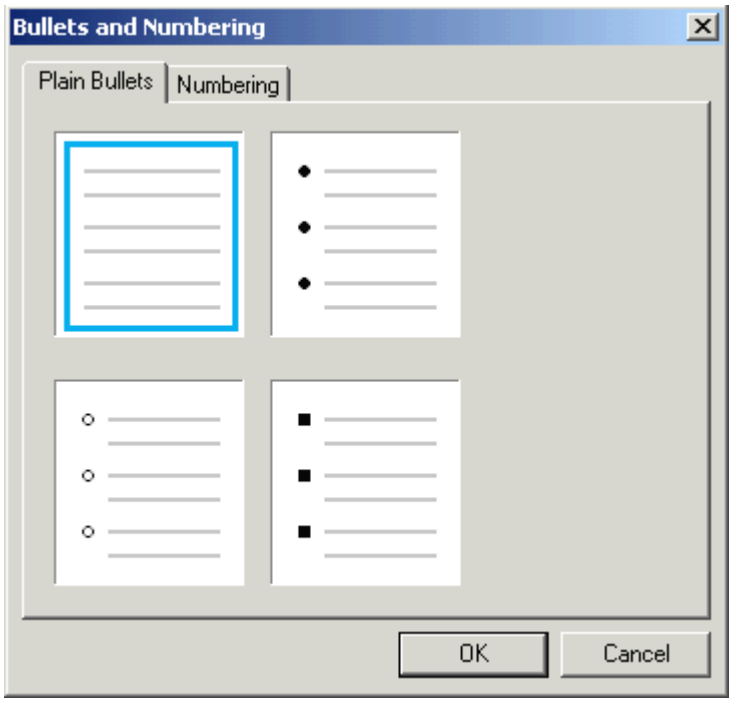
8.5.4 Bullets and Numbering



The **Bullets and Numbering** command allows you to create an element (which has multiple instances) as a list. For example, the persons in a department can be presented in a numbered list, or the paragraphs in a section can be presented as a bulletted list. In each case, it is the element which is to appear as the listitem that must be created as the list.

The Bullets and Numbering command should be used as a StyleVision Power Stylesheet component type rather than inserted around an element or independently of one; that is, an element must be created as a list (when it is dropped into the Design Document), or changed to a list (via the context menu obtained by right-clicking a selected element).

The Bullets and Numbering dialog looks like this:

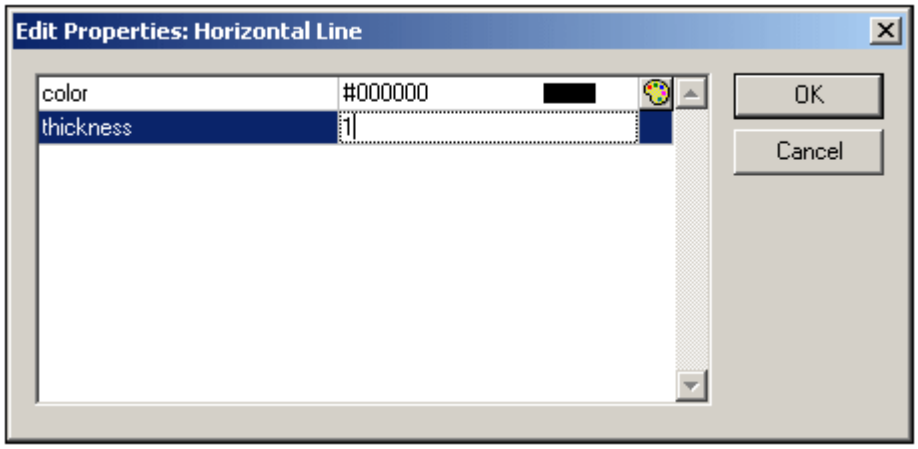


You can specify the required listitem marker in this dialog.

For a description of usage, see Lists in the How to Use section.

8.5.5 Horizontal Line

The **Horizontal Line** command inserts a horizontal rule at the cursor insertion point. It is not available when a StyleVision Power Stylesheet component is selected. By default, the inserted rule is black and has a thickness of 1pt. You can change the default values after inserting the rule by right-clicking the rule and selecting Edit Properties. The following dialog appears:



Edit the color and thickness values as required, and then click OK.

© 2005 Altova GmbH

8.5.6 Contents

The **Contents** command inserts the `(contents)` placeholder for that node. This placeholder represents all the content of the element (text, element, or mixed content), or, in the case of an attribute, its value. The `(contents)` placeholder corresponds to the `xsl:apply-templates` rule of XSLT, which causes the relevant templates to be applied to all child nodes (including attribute and text nodes).

The `(contents)` placeholder can be formatted by selecting it and using a Predefined Format and/or properties in the Text Style and/or Block Style windows. This formatting is visible in the Design Document, and, in the output, it will be applied to the content of the node.

If another node from the schema tree is dropped into a node containing a `(contents)` placeholder, then the existing `(contents)` placeholder is replaced by the new node.

The `(contents)` placeholder can be deleted by selecting it and pressing the Del key on the keyboard. It can also be inserted for a node by placing the cursor inside the node tags, right-clicking, and selecting Insert Contents.

Note: You can create an **empty template rule** by deleting the `(contents)` placeholder of a node. An empty template rule is useful if you wish to define that some node have no template applied to it, i.e. produce no output.

Also see: Processing attribute values for output, Processing element content for output, Data-entry devices, and Rest of Contents.

8.5.7 Rest of Contents

The **Rest of Contents** command inserts the `(rest-of-contents)` placeholder for that node. This placeholder represents the content of **unused child nodes** of the current node; it corresponds to the `xsl:apply-templates` rule of XSLT applied to the unused elements and text nodes of the current element. Note that templates are not applied for child attributes.

Use the `(rest-of-contents)` placeholder in situations where you wish to process one child element in a specific way and apply templates to its siblings. It is important to apply templates to siblings in order to avoid the possibility that the siblings are not processed. This enables you to reach elements lower down in the document hierarchy.

The `(rest-of-contents)` placeholder can be deleted by selecting it and pressing the Del key on the keyboard. It can also be inserted for an element by placing the cursor inside the element tags, right-clicking, and selecting Insert Rest of Contents.

Also see: Contents, Rest-of-Contents, Global Templates.

© 2005 Altova GmbH

8.5.8 Date Picker

The **Date Picker** is a graphical calendar in Authentic View with which the Authentic View user can enter the date for that `xs:date` datatype element. This date is entered in the XML document in the standard XML Schema format for dates: **`YYYY-MM-DD`**..

The Date Picker command allows you to insert a Date Picker at the cursor insertion point. This command is enabled only when the cursor is placed within the tags of an (XML Schema) `xs:date` datatype element and if the element has been created as `(contents)` or an input field.

For a description of how to insert the Date Picker in the StyleVision Power Stylesheet directly the element is created or when an element is changed to `(contents)` or an input field, see Date Picker in the How To section of this documentation.

For the formatting of dates, see Input Formatting.

8.5.9 Page

With the **Page** command you can insert, for paged media output, a page break. These insertions are possible only at cursor insertion points.

Page Break
Click **Page | Break** to define a page break at the cursor insertion point. The page break is displayed as a dashed line across the whole of the Design window. In HTML output, while the page break has no effect in the browser view, a page break will be inserted when the browser view of the HTML file is **printed out**.

Deleting Page Breaks and Page Numbers
To delete page breaks and page numbers, select the placeholder and click **Delete**.

© 2005 Altova GmbH

8.5.10 Bookmark

The **Bookmark (Ctrl+G)** command allows you to insert a bookmark (or anchor) anywhere in the StyleVision Power Stylesheet. A bookmark can be inserted around any StyleVision Power Stylesheet component, or it can be inserted independently of a node. A bookmark can be referenced by a Hyperlink.

To insert a bookmark, do the following:

1. Select any StyleVision Power Stylesheet component or place the cursor at the location where you wish to create the bookmark.
2. Select **Insert | Bookmark**, or right-click and select **Insert | Bookmark**, or use the shortcut CTRL+G.
3. In the Insert Bookmark dialog, enter a name for the bookmark in the Name field. (If there are more bookmarks on the page, they are displayed in the "Other bookmarks on this page" window.)
4. Click OK. The bookmark is defined. If the bookmark is created around a component, then the component is shown with a dashed underline; if the bookmark is free-standing, then it is indicated with a blue flag.

Note:

- A bookmark is created for a single, fixed location in the StyleVision Power Stylesheet and, by extension, in the XML document. If you create a bookmark around an element, the bookmark is created around the first occurrence of that element—and not around each occurrence of that element.
- The name of the bookmark is a static name that you give. It does not come dynamically from the XML document. You cannot assign a dynamic name to a bookmark.

You can delete a bookmark after it has been created by selecting the bookmark in the "Other Bookmarks on this page" list (in the Insert Bookmark dialog), and clicking the Clear button.

Also see: Inserting bookmarks and hyperlinks in the Tutorial section and Hyperlink.

8.5.11 Hyperlink



Ctrl+K

The **Hyperlink** command/icon allows you to insert a link from any part of the output document (HTML) to an anchor within the output document or to an external document or document fragment. Note, however, that linking is not available in Authentic View (since this is a view of the source XML document).

To insert a hyperlink, do the following:

1. Select the StyleVision Power Stylesheet component or text fragment to be made into a hyperlink.
2. Click the Hyperlink icon in the toolbar, or select **Insert | Hyperlink...**, or right-click and select **Insert | Hyperlink...**, or use the shortcut CTRL+K.
3. In the Insert Hyperlink dialog, specify the document or document fragment you wish to link to. You do this by specifying a URL in one of the following forms:
   - a static address (that you enter; you can select an HTML file via the Browse button, and a fragment in the current document via the Bookmark button). Examples would be: `http://www.altova.com` (static Web page URL); `U:\documentation\index.html` (via Browse button); or `#top_of_page` (via Bookmark button).
   - a dynamic address (that comes from a node in the XML document; you specify the node). An example would be a node such as `//otherdocs/doc1`.
   - a combination of static and dynamic text for an address (you specify the static text and the XML document node). An example would be `www.altova.com -- department/name -- #intropara`.
4. Click OK. The hyperlink is created, and the component is underlined to indicate this.

Note: When specifying the node for a dynamic hyperlink entry, you can enter the XPath expression as an absolute XPath by checking the Absolute Path check box. If this check box is not checked, the XPath for the node you select via the Schema button is entered as being relative to the currently selected component.

Using unparsed entities
For the dynamic part of a hyperlink address, you can use the URI declared for an unparsed entity in the DTD—if you are using a DTD. For details of how to use unparsed entities, see Using unparsed entity URIs.

Removing a hyperlink
The link for the selected component can be removed by clicking the Remove Link button in the Insert Hyperlink dialog.

Also see: Inserting bookmarks and hyperlinks in the Tutorial section and Bookmark

 © 2005 Altova GmbH

8.5.12 Auto-Calculation

An **Auto-Calculation** uses an XPath expression to generate a result that is placed in the document at the point where you insert the Auto-Calculation. Note the following points:

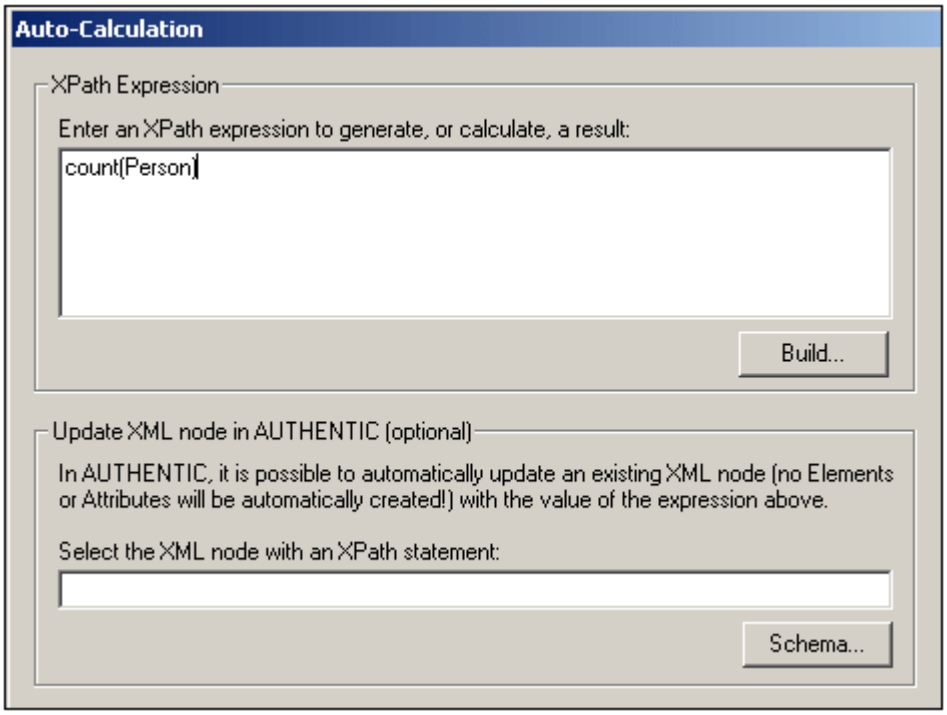
- An Auto-Calculation can be inserted anywhere in the Design Document.
- The XPath version (XPath 1.0 or XPath 2.0) is selected automatically and depends on what XSLT version has been selected for the StyleVision Power Stylesheet. XSLT 1.0 uses XPath 1.0; XSLT 2.0 uses XPath 2.0.
- The point at which you insert the Auto-Calculation determines the context node for the XPath evaluation.
- An Auto-Calculation result is non-editable in Authentic View.
- Any node in the XML document can be updated with the result of the Auto-Calculation.

To insert an Auto-Calculation, do the following:

1. Place the cursor in the Design Document at the point where you wish to insert the Auto-Calculation.
2. Click **Insert | Auto-Calculation**. The following sub-menu appears:



3. Click an option according to what you wish to create the Auto-Calculation as (value, input field, etc). (Note, however, that the Auto-Calculation is non-editable in Authentic View, even if created as an input field.) This pops up the Auto-Calculation dialog:


Auto-Calculation
XPath Expression
Enter an XPath expression to generate, or calculate, a result:
count(Person)
Build...
Update XML node in AUTHENTIC (optional)
In AUTHENTIC, it is possible to automatically update an existing XML node (no Elements or Attributes will be automatically created!) with the value of the expression above.
Select the XML node with an XPath statement:
Schema...

4. Build or enter the XPath expression.

5. If you wish to update a node, select the node by clicking the Schema... button and then the required node.

For examples of how to use Auto-Calculations, see Using Auto-Calculation.

© 2005 Altova GmbH

8.5.13 Condition

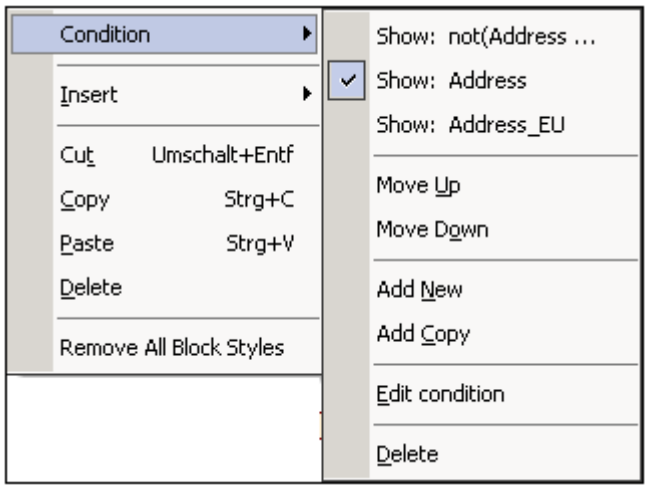
The **Condition...** command enables you to insert a condition (in a conditional template) at the cursor point or around the selection. A conditional template consists of one or more conditions, and each condition has specific processing rules associated with it.

A condition can be inserted at any point in the StyleVision Power Stylesheet. The first condition you insert creates the conditional template; subsequently, conditions can be added to the same conditional template. Use the **Insert | Condition...** command to insert a condition.

The location in the StyleVision Power Stylesheet determines the context node for the XPath expression used to specify the condition. When you specify the XPath expression, you can specify it as an absolute XPath or relative to the context node.

A node is tested against each condition in turn. The first condition that is evaluated as being true for a node causes the processing rules associated with that condition to be applied to that node; the conditional template is then exited. If no condition is evaluated as true, either a default processing—if you have defined one—is used for the node, or no processing is used at all. The order in which conditions are specified is, therefore, significant.

Right-clicking a condition causes the following sub-menu for the Condition command to appear:



The currently selected (displayed) condition is indicated with a check mark. The Edit condition command allows you to edit the XPath expression that defines this condition. To edit another condition, you have to first select the condition you wish to edit, and then select the Edit condition command. You can move the currently selected condition up or down the list of conditions in the conditional template by clicking the Move Up or Move Down commands. The relative position of conditions within a conditional template is significant because the first condition to evaluate as true is executed and the conditional template is then exited.

The use of conditional templates is described with examples in Conditional templates in the How To Use section.

8.6 Table

The **Table** menu provides commands enabling you to insert a static table and to change the structure and properties for both static and dynamic tables. You can edit table structure by appending, inserting, deleting, joining, and splitting rows and columns. Properties of the table as well as of individual columns, rows, and cells are defined in the Table Properties dialog.

The Table commands are available in the Table menu and as icons in the toolbar. The availability of various table commands depends on the current cursor position. A static table can be inserted at any location in the StyleVision Power Stylesheet by clicking the Insert Table command. A dynamic table is inserted by creating an element as, or changing an element to, a table. To edit the table structure, place the cursor in the appropriate cell, column, or row, and select the required editing command. To edit a formatting property, place the cursor in the appropriate cell, column, row, or table, and define the required property in the Table Properties dialog.

Headers and footers
When you create a dynamic table, you can specify whether you wish to include headers and/or footers. (Footers are allowed only when the table grows top–down.) You can create a header and footer in a static table by manually inserting a top and bottom row, respectively. The structures of headers and footers in both static and dynamic tables can be modified by splitting and joining cells.

Navigating in tables
Use the Tab and arrow keys to navigate the table cells.

Adding cell content
You can inset any type of StyleVision Power Stylesheet component as the content of a cell. The component should be formatted using the standard formatting tools.

See: Tables for a description of the various types of tables used in StyleVision Power Stylesheets, and CALS / HTML Table properties... for a description of how to enable XML tables for the Authentic View user.

© 2005 Altova GmbH

8.6.1 Insert Table...



The **Insert Table...** command inserts an empty static table into the design tab. Selecting this command opens a dialog box which allows you to define the size of the table (in terms of its rows and columns). You can change this structure subsequently by appending, inserting, and deleting rows and/or columns.

To insert a dynamic table, see Creating dynamic tables.

8.6.2 Delete Table



The **Delete Table** command deletes the static or dynamic table in which the cursor is.

© 2005 Altova GmbH

8.6.3 Append Row



The **Append Row** command appends a row to the static or dynamic table in which the cursor is.

8.6.4 Append Column



The **Append Column** command appends a column to the static or dynamic table in which the cursor is.

© 2005 Altova GmbH

8.6.5 Insert Row



The **Insert Row** command inserts a row above the row in which the cursor is. This command applies to both static and dynamic tables.

8.6.6 Insert Column



The **Insert Column** command inserts a column to the left of the column in which the cursor is. This command applies to both static and dynamic tables..

© 2005 Altova GmbH

8.6.7 Delete Row



The **Delete Row** command deletes the row in which the cursor is. This command applies to both static and dynamic tables.

8.6.8 Delete Column



The **Delete Column** command deletes the column in which the cursor is. This command applies to both static and dynamic tables.

© 2005 Altova GmbH

8.6.9 Join Cell Left



The **Join Cell Left** command joins the cell in which the cursor is to the adjacent cell on the left. The contents of both cells are concatenated in the new cell. All property values of the cell to the left are passed to the new cell. This command applies to both static and dynamic tables.

8.6.10 Join Cell Right



The **Join Cell Right** command joins the cell in which the cursor is to the cell on the right. The contents of both cells are concatenated in the new cell. All property values of the cell to the left are passed to the new cell. This command applies to both static and dynamic tables.

© 2005 Altova GmbH

8.6.11 Join Cell Below



The **Join Cell Below** command joins the cell in which the cursor is to the cell below. The contents of both cells are concatenated in the new cell. All property values of the cell on the top are passed to the new cell. This command applies to both static and dynamic tables.

8.6.12 Join Cell Above



The **Join Cell Above** command joins the cell in which the cursor is to the cell above. The contents of both cells are concatenated in the new cell. All property values of the cell on top are passed to the new cell. This command applies to both static and dynamic tables.

© 2005 Altova GmbH

8.6.13 Split Cell Horizontally



The **Split Cell Horizontally** command creates a new cell to the right of the cell in which the cursor is. The contents of the original cell stay in the original cell. All properties of the original cell are passed to the new cell. This command applies to both static and dynamic tables.

8.6.14 Split Cell Vertically



The **Split Cell Vertically** command creates a new cell below the cell in which the cursor is. The contents of the original cell remain in the upper cell. All properties of the original cell are passed to the new cell. This command applies to both static and dynamic tables.

© 2005 Altova GmbH

8.6.15 View Cell Bounds

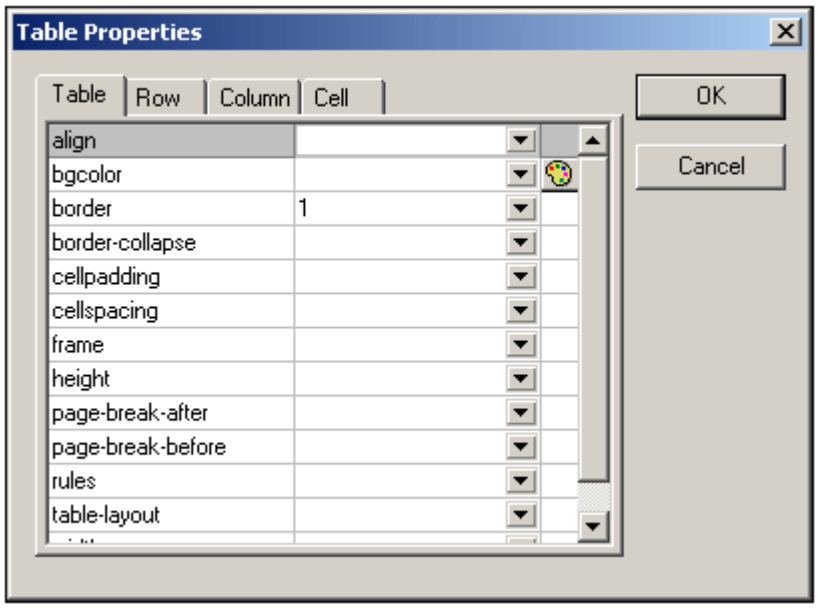


The **View Cell Bounds** command toggles the display of table boundaries (borders) on and off for tables that have a table border value of 0. This command applies to both static and dynamic tables.

8.6.16 Table Properties



The **Table Properties** command opens the Table Properties dialog box for static and dynamic tables.



In the Table Properties dialog, you can specify the formatting properties of the table as a whole, as well as for individual rows, columns, and cells. The cursor must be in the cell, column, row, or table to be formatted before you select this command. The table properties available in this dialog closely follow those of HTML. So, the `frame` and `rules` properties define whether and how table and cell borders are drawn, `border` defines the width of borders, and so on.

Cellpadding
The following points about cellpadding in tables should be noted:

- In the Table Properties dialog, cellpadding is specified globally in the Table tab as the padding of all cells in the table. To specify padding for a particular cell, place the cursor in the cell and specify the padding (separately for each of the four components: left, top, right, and bottom) in the Layout tab of the Block Style window.

© 2005 Altova GmbH

8.6.17 Vertical alignment of cell content

Commands to set the vertical alignment of cell content are available as icons in the toolbar. Place the cursor anywhere in the cell, and click the required icon.



Vertically Align Top vertically aligns cell content with the top of the cell.



Vertically Align Middle vertically aligns cell content with the middle.



Vertically Align Bottom vertically aligns cell content with the bottom of the cell.

8.7 Authentic

The **Authentic** menu contains commands that enable you to customize aspects of the Authentic View of an XML document that will be displayed using the StyleVision Power Stylesheet. The document processing that you specify using commands in this menu apply only for Authentic View; they do not apply for HTML output.

© 2005 Altova GmbH

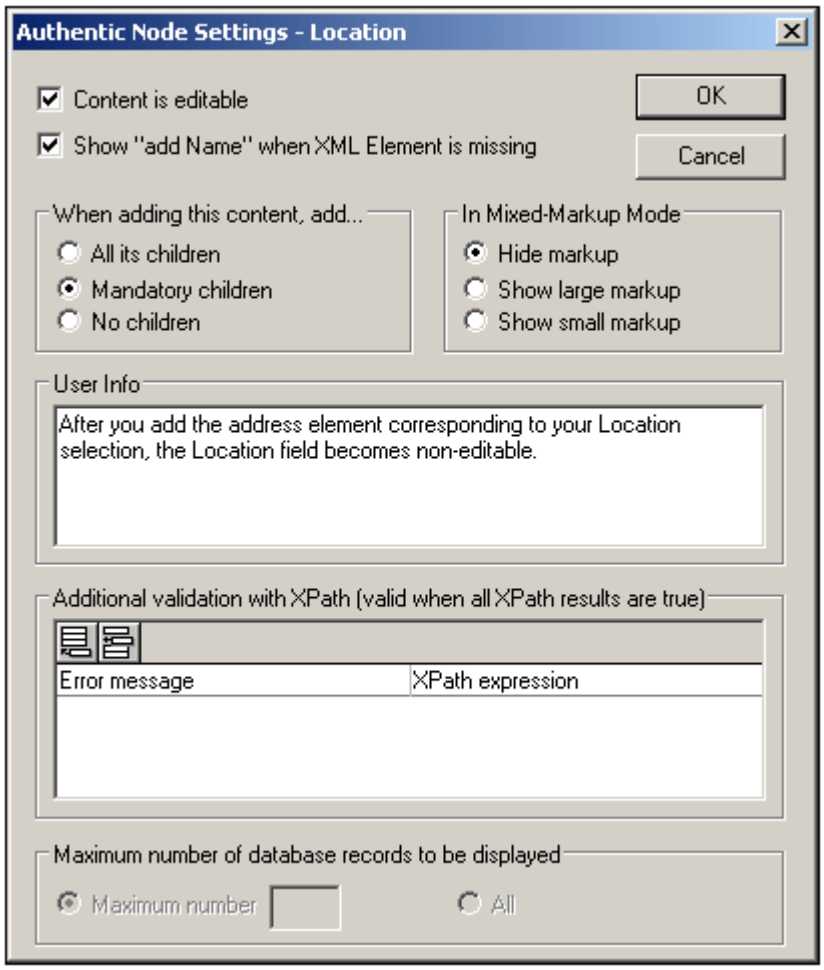
8.7.1 Node Settings...



The **Node Settings...** command allows you to define the properties of a node for display **in Authentic View**. You can specify node settings for elements, attributes, the text content of elements and attributes, and Auto-Calculations. Select the element/attribute tag to assign node settings for the element or attribute. Select the `(contents)` placeholder, data-entry device, etc, or Auto-Calculation to assign node settings for text content and Auto-Calculations. If node settings have been specified for both an element/attribute node as well as for its contents, then the node settings for the contents takes precedence.

You can also call the Authentic Node Settings dialog by right-clicking the node and selecting **Authentic Node Settings...**.

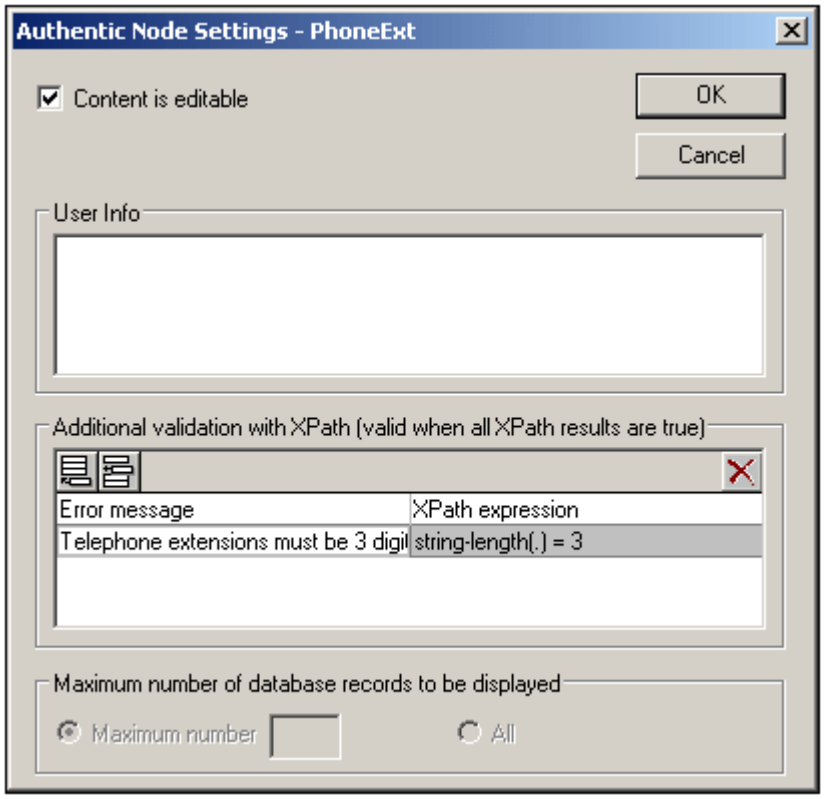
Settings for Element/Attribute nodes
The following Authentic Node Settings dialog appears when an element or attribute is selected.


Authentic Node Settings - Location
Content is editable
Show "add Name" when XML Element is missing
OK
Cancel
When adding this content, add...
All its children
Mandatory children
No children
In Mixed-Markup Mode
Hide markup
Show large markup
Show small markup
User Info
After you add the address element corresponding to your Location selection, the Location field becomes non-editable.
Additional validation with XPath (valid when all XPath results are true)
Error message
XPath expression
Maximum number of database records to be displayed
Maximum number
All

The name of the selected element/attribute appears in the title bar of the dialog.

Settings for content nodes and Auto-Calculations

The following Authentic Node Settings dialog appears when the `(contents)` placeholder, data-entry device, etc, or an Auto-Calculation is selected.



The name of the parent element/attribute appears in the title bar of the dialog.

Authentic Node Settings
The following node settings can be made to control the behavior of individual nodes in the Authentic View display.

Content is editable
Checking this box enables the Authentic View user to edit the XML content of the node. The check box is checked by default. This setting is available when the selected node is an element, attribute, or contents. Auto-Calculation results cannot be edited; this value is computed with the XPath expression you enter for the Auto-Calculation.

Show "add Name" when XML Element is missing
Checking this check box specifies that a prompt ("Add [element/attribute name]") will appear in Authentic View when the selected element or attribute is missing. The check box is checked by default. This setting is available when the selected node is an element or an attribute.

When adding content, add...
This setting is available when the selected node is an element. It allows you to define what child elements of the selected element are inserted when the selected element is added. The options are: all child elements, mandatory child elements, and no child element.

In Mixed-Markup Mode
This setting is available when the selected node is an element or attribute, and enables you to specify how individual nodes will be marked up in the mixed markup mode of Authentic View. The following options exist: large markup (tags with node names); small markup (tags without node names); and no markup.

User Info

 © 2005 Altova GmbH

Text entered in this text box appears as a tooltip when the mouse pointer is placed over the node. It is available when the selected node is an element, attribute, contents, or an Auto-Calculation. If both the element/attribute node as well as the contents node has User Info, then the User Info for the contents node is displayed as the tooltip when the mouse is placed over the node.

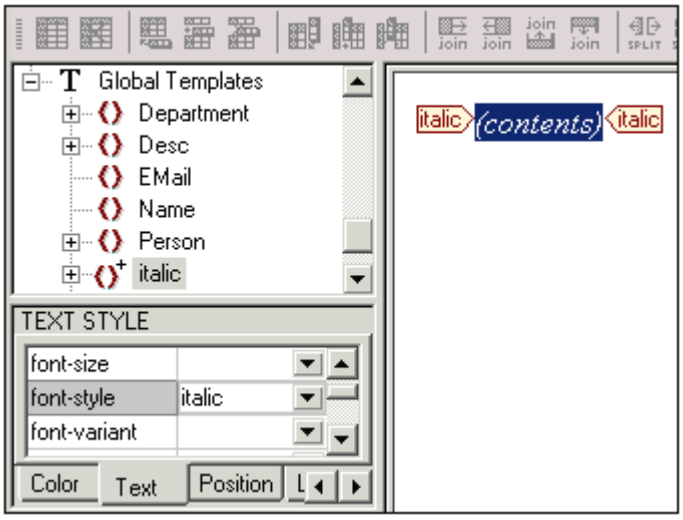
Additional validation with XPath
This setting is available when the selected node is an element, attribute, or contents. You can set an XPath expression to define the validity of the node's XML value. An XML value that falls outside this defined range is invalid. If the XML value of the node is invalid, this is made known to the Authentic View user by means of an **error message** when the XML document is validated (**F8**). The error message that is displayed is the text you enter into the Error message field of the Additional Validation setting. Additionally, if the selected node is contents, invalid data is **displayed in red** in Authentic View, and the **error message** is displayed on a mouseover.

8.7.2 Text State Icons...

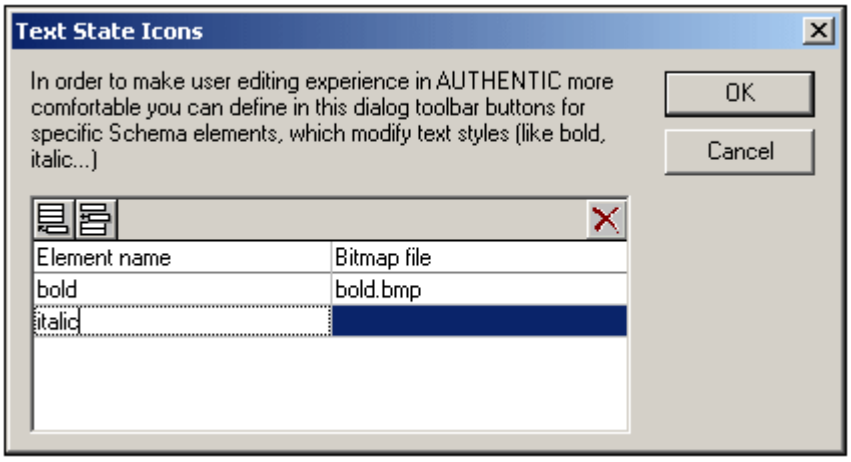
The **Text State Icons...** command enables you to define an icon for a global element. Once defined the icon can be displayed in the toolbar of Authentic View (i.e. in Altova's XMLSpy 2005, Authentic 2005 Desktop Edition, and Authentic View products), thus allowing the Authentic View user to insert an element around selected text by clicking the icon. This feature is intended for elements that provide inline formatting, such as bold and italic fonts.

All global elements in the schema appear in the Global Templates list in the Schema Window. To create a Text State icon, do the following:

1. Define a global template for the global element that is to get a Text State icon. The formatting that you define for the global template will be applied to text when the Authentic View user clicks the Text State icon. The screenshot below shows the global template for the `italic` element. (If the element you require is not available as a global element, you must make it a global element if you wish to use this feature.)

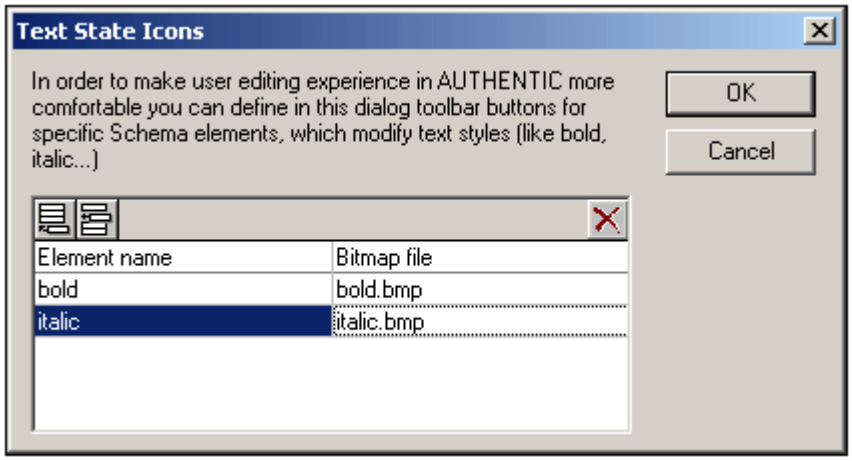


2. Select the menu option **Authentic | Text State Icons...**. This opens the Text State Icons dialog box.
3. Click the Append button to add a new Text State icon.
4. In the **Element name** field, enter the name of the element for which you wish to create the Text State icon, and click OK.


Text State Icons
In order to make user editing experience in AUTHENTIC more comfortable you can define in this dialog toolbar buttons for specific Schema elements, which modify text styles (like bold, italic...)
OK
Cancel
Element name
Bitmap file
bold
bold.bmp
italic

5. Press **Tab** or double click in the **Bitmap file** column, and enter the name of the icon

image that you wish to have displayed in the Authentic View toolbar.


Text State Icons
In order to make user editing experience in AUTHENTIC more comfortable you can define in this dialog toolbar buttons for specific Schema elements, which modify text styles (like bold, italic...)
OK
Cancel
Element name
Bitmap file
bold
bold.bmp
italic
italic.bmp

6. Click OK to confirm.
7. Place the icon image file (`.bmp` file) in the `\\sps\Picts` folder of your application folder.

When you edit an XML document in Authentic View (in Altova's XMLSpy 2005, Authentic 2005 Desktop Edition, and Authentic View products) using this StyleVision Power Stylesheet, the icon image appears in the toolbar as a Text State icon. If the Authentic View user selects text and clicks the icon, the element represented by that icon is created around the selected text and the formatting you defined in the global template for that element is applied to the selected text.

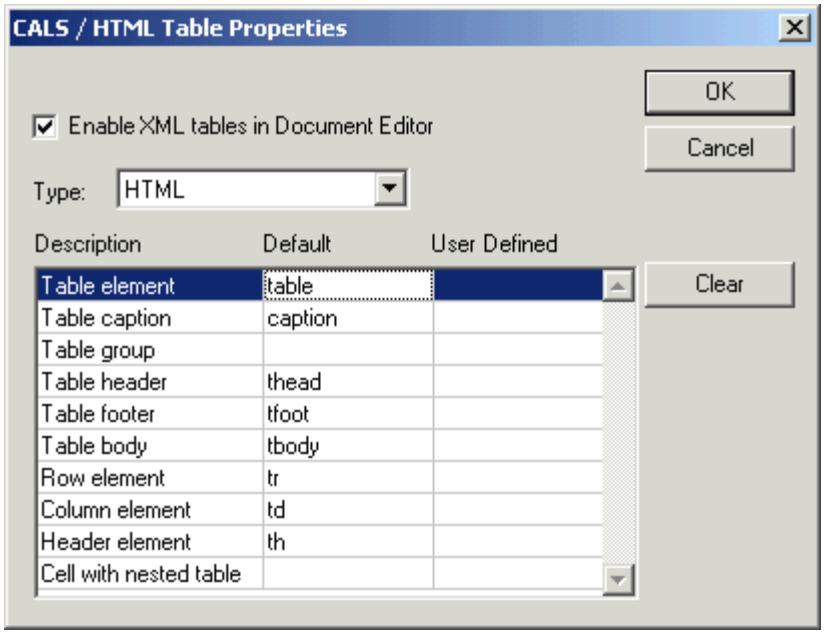
Note: Text State icons are not available in Authentic Preview of StyleVision.

8.7.3 CALS / HTML Tables...

The **CALS / HTML Tables...** command pops up a dialog in which you specify:

- whether the Authentic View user may insert XML tables (following either the CALS or the HTML table model) in the XML document or not, and
- the XML elements that will constitute the XML table structure if XML tables are allowed.

When you click **Authentic | CALS / HTML Tables...**, the following dialog appears:



Enabling XML Tables in Authentic View
To enable XML tables in Authentic View, check the box marked Enable XML tables in Document Editor.

Enabling XML tables allows the Authentic View user to insert tables that can be structured and formatted as required by the user. This is as opposed to static and dynamic SPS tables, which are structured and formatted by you, the designer of the StyleVision Power Stylesheet. When an XML table is inserted in Authentic View, a set of XML elements conforming to either the CALS or HTML table model is inserted in the XML document. Cell content in Authentic View is entered as content of the corresponding element in the XML document. When the table is formatted in Authentic View, the appropriate attributes and their values are added to the relevant elements in the XML document.

Defining schema elements for the selected table model
In the combo box, select the table model (CALS or HTML) with which you want your schema table structure to correspond. This combo box is enabled only if the Enable XML tables check box has been checked. Note the following points:

- The **table elements in the schema** must correspond exactly with the structure of the selected table model (either CALS or HTML) to ensure the correct functioning of this feature; however, they do not need to have the same names as those of the table model elements.
- If a table element in the schema is named differently from the corresponding element in the selected table model, then the schema element must be mapped to the corresponding table model element by entering its name in the User Defined column of the CALS / HTML Table Properties dialog.

 © 2005 Altova GmbH

- The names of all **attributes** of schema table elements must correspond exactly with the relevant attribute names in the table model. These attributes must therefore be defined in the schema. Attribute names must correspond exactly because when the Authentic View user modifies the table structure or enters (for HTML tables) table properties, Authentic View enters attributes and enters/modifies attribute values for the appropriate element in the XML file. If these attributes are not present in the schema or are named (or spelled) differently in the schema, then the XML file will be invalid.

After selecting the appropriate model, all elements in the selected model which do not exist in the table structure of the schema, or which do not have a corresponding schema element specified for it, are displayed in red. If a schema element exists that corresponds to a CALS/HTML element, then you must map this element to the corresponding CALS/HTML element.

Note: When all the elements in the CALS / HTML Table Properties dialog are displayed in black, this means that the schema contains elements that correspond to the elements of the selected table model. If an element name in the Default column is displayed in red, this indicates that either no corresponding schema element exists or that a corresponding schema element exists but has not been mapped to the table model element; in the latter case you must enter the name of the corresponding schema element into the User Defined column.

Caution: If all the element names are displayed in black, be aware that this does not necessarily mean that the table structure of the schema corresponds exactly with the selected table model. There may still be additional elements in the table structure of the schema that could cause errors. You should carefully check your schema for this. Also check the attributes of the schema elements for exact correspondence and naming. Absent or wrongly named attributes will cause validation errors if the Authentic View user uses table formatting properties that use these attributes.

Also see: Tables: XML Tables in the How To Use section of this documentation.

8.7.4 Assign Template XML File...

The **Assign Template XML** command assigns an XML file to the StyleVision Power Stylesheet (SPS). When the SPS is opened in Authentic View, it displays the data present in the assigned XML file according to the design of the SPS. The XML file therefore provides the starting data of a new XML file. In effect it provides the data for an template XML file, which is based on the SPS and can be saved under any name. We therefore call the XML file you assign to the SPS a Template XML File.

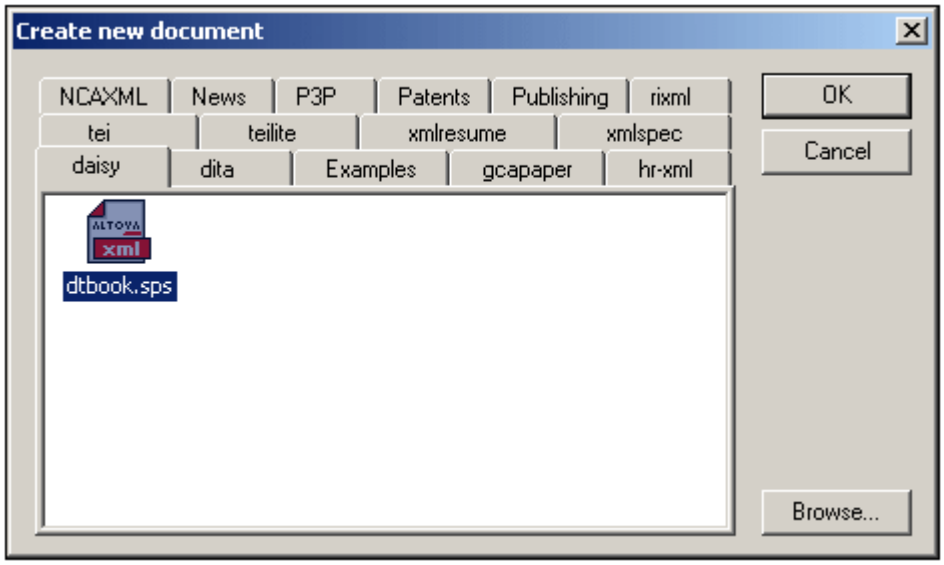
Note: The Template XML file must conform to the same schema as that of the StyleVision Power Stylesheet to which it is linked.

Assigning a Template XML File
To assign a Template XML File, click **Authentic | Assign Template XML File...**, select the required file, and save the StyleVision Power Stylesheet.

Changing the Template XML File
To change the Template XML File, click **Authentic | Assign Template XML File...**, select the new Template XML File, and save the StyleVision Power Stylesheet.

Opening a template XML document in Authentic View
An Authentic View user can open a template XML document as follows:

1. In XMLSpy 2005 or Authentic 2005 Desktop Edition, select **File | New** or **Authentic | New Document...**. This pops up the Create New Document dialog. The tabs in this dialog each represent a folder in the `sps/Template` folder within the XMLSpy 2005 or Authentic 2005 Desktop Edition application folder. In each tab, the SPS files for the corresponding folder are displayed. These SPS files provide general use templates for popular schemas.



2. Select an Altova-supplied SPS template from one of the tabs or browse for the required StyleVision Power Stylesheet (using the **Browse...** button).
3. Click OK. A template based on the SPS and carrying starting data from a Template XML File, if any is associated with the SPS, is opened in Authentic View.

Adding folders and SPS files to the tabbed list in the Create a New Document dialog
You can add your own SPS files to the tabbed list in the Create a New Document dialog. Do this by copying/moving the folders containing the SPS files into the `sps/Template` folder within the

 © 2005 Altova GmbH

XMLSpy 2005 or Authentic 2005 Desktop Edition application folder. The added folder will then be displayed as a tab in the Create a New Document dialog. The SPS files which are in the folder will be displayed in the tab for that folder.

Unassigning the Template XML File

Open the StyleVision Power Stylesheet in an XML editor application, such as XMLSpy 2005, or in a plain text editor, such as Notepad. The root element of the StyleVision Power Stylesheet is `<structure>`. Delete the contents of the `templatexmlfile` attribute of `structure`, so that the attribute has an empty value. The start tag of the `structure` element should now look something like this:

```
<structure version="2" schemafile="test.xsd" workingxmlfile=""
templatexmlfile="">
```

Save the StyleVision Power Stylesheet. The Template XML File is now unassigned.

8.7.5 Auto-add Date Picker



This is a toggle command that switches the Auto-add Date Picker ON and OFF. When the Auto-add Date Picker is ON, any `xs:date` datatype element that is created as contents or as an input field will have the Date Picker automatically inserted within the element tags and after the `(contents)` placeholder or input field.

Also see: Date Picker in the How To Use section.

© 2005 Altova GmbH

8.7.6 Table

The **Table** command allows you to manipulate the rows of a **dynamic table**. You can append, insert, duplicate, and delete rows, and you can move the selected row up and down relative to the other rows of the table. A row is selected by placing the cursor inside it.

Note: Each row of a dynamic table represents an occurrence of a repeatable element within the set of all those repeatable elements.

8.7.7 Markup

With the Markup command, you can select between the Hide Markup and Show Full Markup modes. In Hide Markup mode, node tags are not displayed. In Show Full Markup mode, opening and closing tags, with their node names, are displayed.

© 2005 Altova GmbH

8.7.8 Define Entities

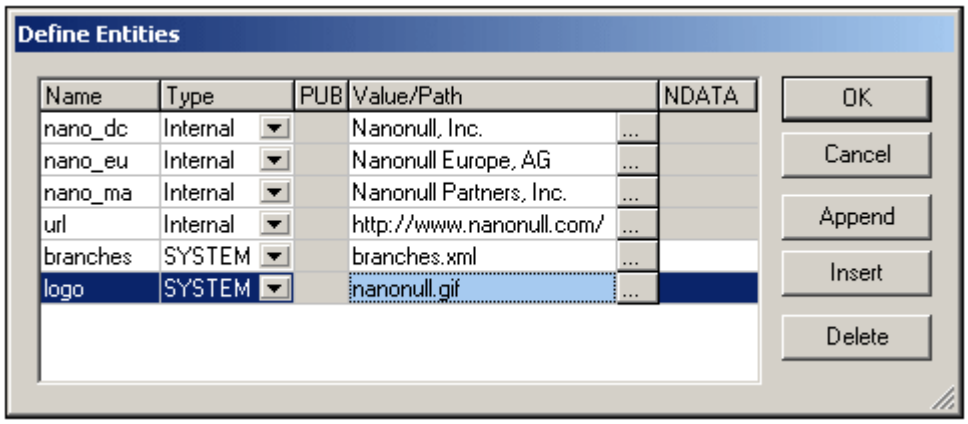


The **Define Entities** command is available only in Authentic View. With the **Define Entities** command in Authentic View, you can define entities that you want to add to your **XML document**. After an entity has been defined, it can be inserted in the XML document by right-clicking at the location where you wish to insert the entity, and, from the context menu that pops up, selecting **Insert Entity**, and then the name of the entity to be inserted.

An entity that you define with this command can be any of three types:

- Internal parsed entity. The value of the entity is a text string that usually occurs frequently in the document. Using an entity ensures that all occurrences are expanded to the value defined here.
- External parsed entity. This is an external XML file that will replace each occurrence of the entity. The value of the entity is the URI of the external XML file.
- External unparsed entity. This is an external resource that will be called when the entity is processed. The value of the entity is the URI of the external resource.

Clicking the command, pops up the Define Entities dialog (shown below).



For a description of how to use this dialog, see Define Entities in the Authentic View documentation.

8.7.9 Validate XML

 **F8**

This command checks the validity of the XML file against the associated schema. Any additional validation requirement that you have entered for individual nodes (**Authentic | Node Settings**) is also checked. The result of the validation check is displayed in a pop-up message box.

© 2005 Altova GmbH

8.8 HTML Import

The **HTML Import** menu enables you to convert HTML files to XML, and to auto-generate the XML Schema/DTD, XSLT stylesheet, and XML content files. For a detailed explanation, see the tutorial, Converting HTML to XML.

8.8.1 Import HTML file...

The **Open HTML file...** command enables you to open an HTML file in StyleVision. On clicking this command, a dialog opens in which you can browse for the HTML file you wish to import and convert to XML. Having selected the file you want to import, it then appears in the Design Document window. You then drag the elements you want to convert to XML into the schema window at left.

Please see the HTML to XML Converter documentation for details.

© 2005 Altova GmbH

8.8.2 Add to Parent

Adds the selected item, in the Design window, as a **child item** to the currently active element/attribute in the Schema window.

1. Select the XML **parent item** by clicking it in the schema window (not necessary if the root element is the only visible element in the schema view).
2. Select the HTML item (e.g. header or text) you want to define as the child item (in the **Design [Document]** window), and
3. Select the specific menu item from the list below.

Convert selection to elements
Converts the HTML selection(s) in the Design [Document] window, into an XML element.

Convert selection to attributes
Converts the HTML selection(s) in the Design [Document] window, into an XML attribute.

Surround selection with element
Embeds the current HTML selection between an XML start and XML end tag.

Convert selected Table/List to elements
Converts the HTML selection (even if the text cursor is only set in a table), into an XML table where all HTML items are converted into XML elements.

Convert selected Table/List to attributes
Converts the HTML selection (even if the text cursor is only set in a table), into an XML table where all HTML items are converted into XML attributes.

8.8.3 Insert

Inserts the selected item, in the Design window, **above** the currently active element/attribute in the Schema window.

1. Select/click the XML item **above** which the element/attribute is to be inserted in the **schema window**.
2. Select the HTML item (e.g. header or text) you want to insert and convert to XML, in the **Design [Document]** window.
3. Select the specific menu item from the list below.

Convert selection to elements
Converts the HTML selection(s) in the Design [Document] window, into an XML element.

Convert selection to attributes
Converts the HTML selection(s) in the Design [Document] window, into an XML attribute.

Surround selection with element
Embeds the current HTML selection between an XML start and XML end tag.

Convert selected Table/List to elements
Converts the HTML selection (even if the text cursor is only set in a table), into an XML table where all HTML items are converted into XML elements.

Convert selected Table/List to attributes
Converts the HTML selection (even if the text cursor is only set in a table), into an XML table where all HTML items are converted into XML attributes.

© 2005 Altova GmbH

8.8.4 Append

Appends the selected item, in the Design window, to the currently active element/attribute in the Schema window.

1. Select an XML item to which the element/attribute is to be appended in the **schema window**.
2. Select the HTML item (e.g. header or text) you want to append and convert to XML, in the **Design [Document]** window.
3. Select the specific menu item from the list below.

Convert selection to elements
Converts the HTML selection(s) in the Design [Document] window, into an XML element.

Convert selection to attributes
Converts the HTML selection(s) in the Design [Document] window, into an XML attribute.

Surround selection with element
Embeds the current HTML selection between an XML start and XML end tag.

Convert selected Table/List to elements
Converts the HTML selection (even if the text cursor is only set in a table), into an XML table where all HTML items are converted into XML elements.

Convert selected Table/List to attributes
Converts the HTML selection (even if the text cursor is only set in a table), into an XML table where all HTML items are converted into XML attributes.

8.8.5 Settings...

The "Settings..." command allows you to define the specific HTML import settings.



Import whitespace text:
Imports and preserves the whitespace of imported text.

Import empty cells from table/list:
Imports empty cells from HTML tables or lists.

© 2005 Altova GmbH

8.9 Properties

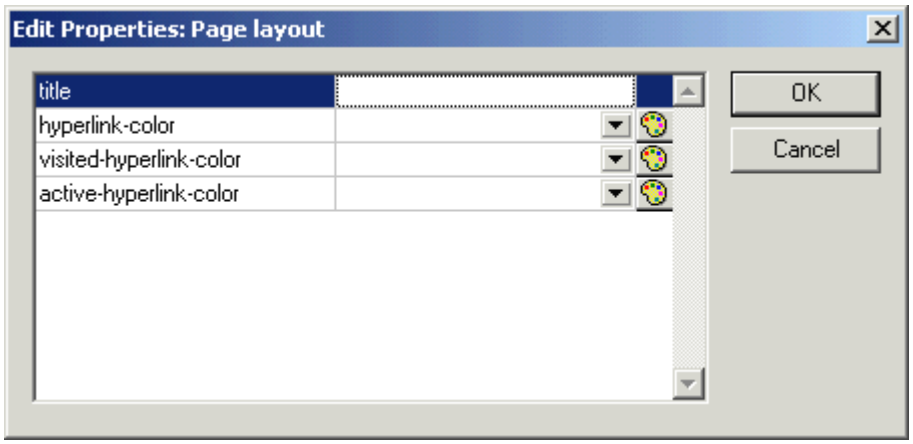
The **Properties** menu allows you to edit the properties of:

- the HTML output page, and
- a variety of StyleVision Power Stylesheet components, such as tables, lists, and data-entry devices

These properties are described in the following sections.

8.9.1 HTML Page...

The **HTML Page...** command allows you to define certain properties of the output HTML page. These are the page title, which appears in the title bar of the HTML page, and the colors of the links in the document.



Note: The properties defined here take precedence over the individual HTML element settings.

© 2005 Altova GmbH

8.9.2 Table...

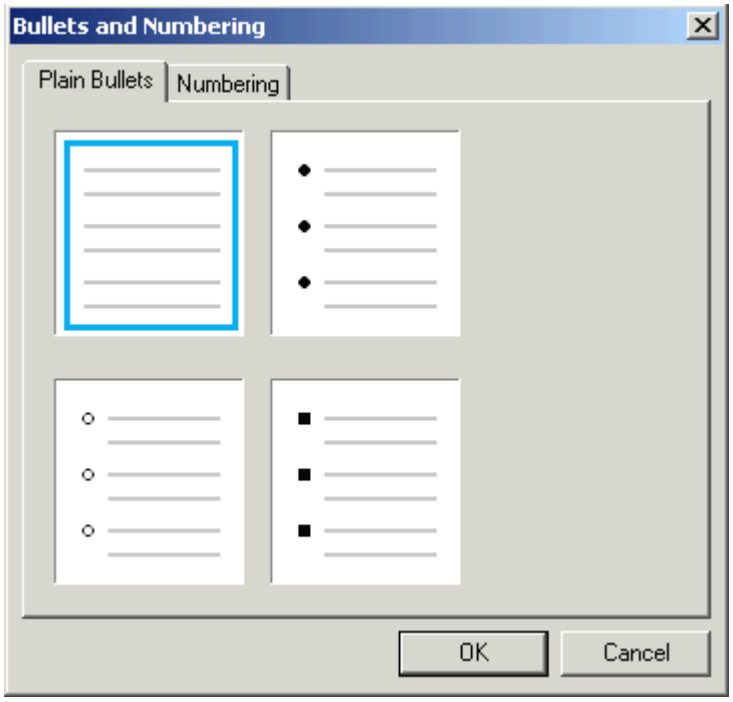


The **Table...** command opens the Table Properties dialog box, which allows you to define specific properties of static and dynamic tables. See Formatting static and dynamic tables and Table Properties for details.

8.9.3 Bullets and Numbering...



The **Bullets and Numbering...** command allows you to **edit** the bullet and numbering styles of the selected list; in the case of a numbered list, the initial number can also be specified.



Note: This command should be used to change the list style of an already created list.

Also see: Lists and Bullets and Numbering.

© 2005 Altova GmbH

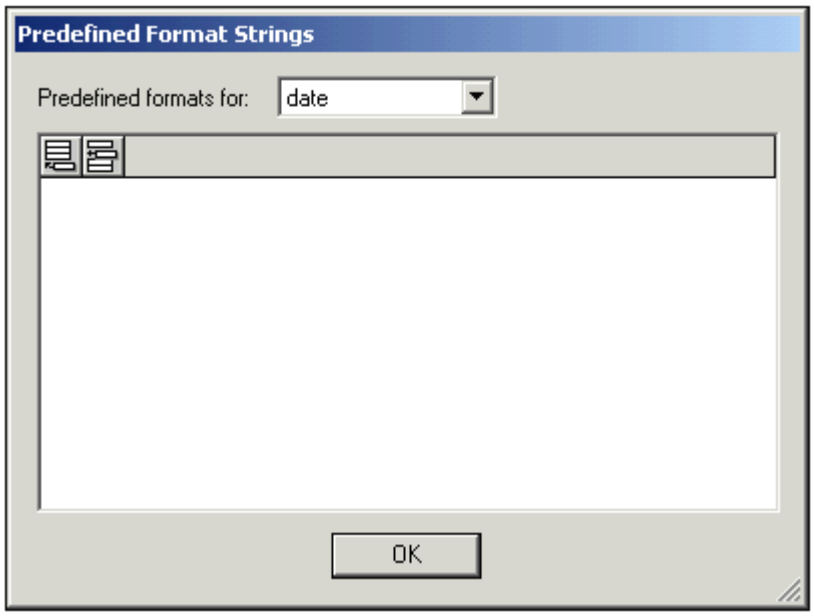
8.9.4 Predefined Format Strings

Any `(contents)` placeholder, input field, or Auto-Calculation which is of a `numeric`, `date`, `time`, `dateTime` or `duration` datatype can be assigned a custom format with the Input Formatting dialog. In the Input Formatting dialog, you can either create a format directly or select from a drop-down list of predefined formats. This list consists of two types of predefined formats:

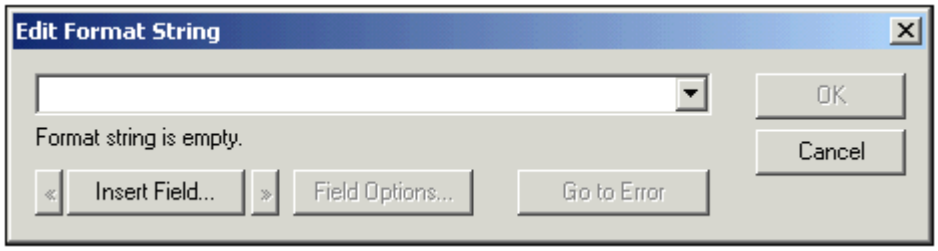
- supplied predefined formats (delivered with StyleVision), and
- customized predefined formats that you define with the **Predefined Format Strings** command. These customized formats are created for the currently open SPS file—and not for the entire application. After the customized formats have been defined, the SPS File must be saved in order for the formats to be available when the file is next opened.

Creating a predefined format string
A predefined format string is specific to a datatype. To create a predefined format string, do the following:

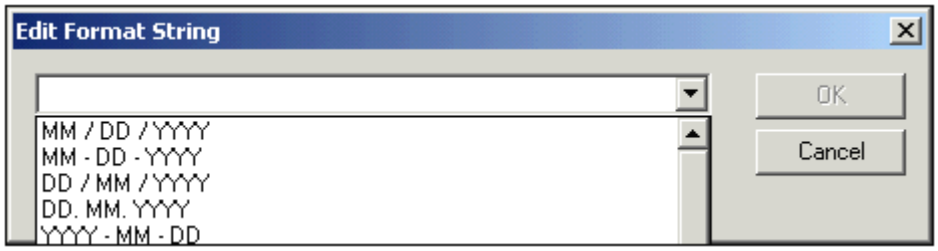
1. Click **Properties | Predefined Format Strings...**. The following dialog appears:



2. Select a datatype from the drop-down list in the combo box, and then click the Append or Insert icon as required. This pops up the Edit Format String dialog:



If you click the down arrow of the combo box, a drop-down list with the supplied predefined formats for that datatype is displayed (shown in the screenshot below).



You can either select a format from the list and modify it, or you can enter a format directly into the input field. The syntax for defining a format is explained in Input Formatting. If you need help with the syntax, use the **Insert Field...** and **Field Options...** buttons.

3. After you have defined a format, click OK. The format string is added to the list of predefined formats for that datatype, and it will appear as an option in the Input Formatting dialog (of the current SPS file) when the selected element is of the corresponding datatype.

Note:

- You can add as many custom format strings for different datatypes as you want.
- The sequential order of format strings in the Predefined Format Strings dialog determines the order in which these format strings appear in the Input Formatting dialog. The customized format strings appear above the supplied predefined formats.
- To edit a custom format string, double-click the entry in the Predefined Format Strings dialog.
- To delete a custom format string, select it, and click the Delete button in the dialog.

© 2005 Altova GmbH

8.9.5 Selected object...

The **Selected object...** command enables you to edit the properties of the selected StyleVision Power Stylesheet component. It is enabled when a data-entry device or a horizontal rule is selected. When you click this command, the properties dialog of the selected component type opens. Enter the required value and click OK.

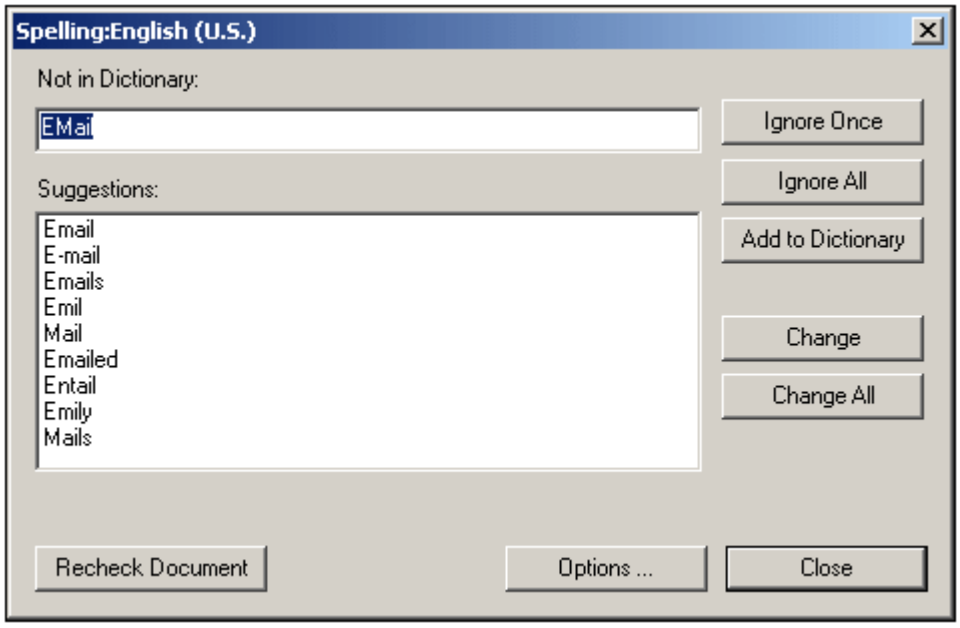
8.10 Tools

The **Tools** menu contains the Spell-checker command and commands that enable you to customize StyleVision.

© 2005 Altova GmbH

8.10.1 Spelling...

The **Spelling... (Shift+F7)** command runs a spell check on the StyleVision Power Stylesheet. The dialog shown below appears. Words that are not present in the selected dictionary are displayed in the Not in Dictionary field of the dialog and highlighted in the Design Document.



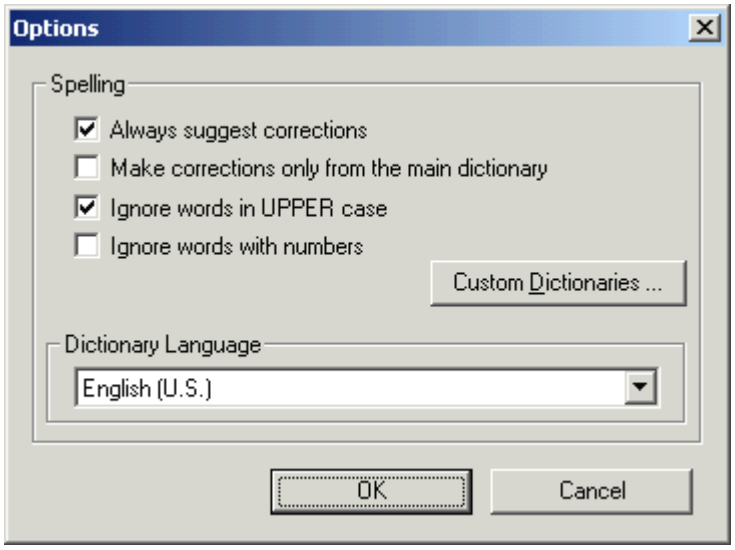
You can then select an entry from the list in the Suggestions pane and click **Change** or **Change All** to change the highlighted instance of this spelling or all its instances, respectively. (Double-clicking a word in the Suggestions list causes it to replace the unknown word.) Alternatively, you can ignore *this instance* of the unknown word (**Ignore Once**); or ignore *all instances* of this unknown word (**Ignore All**); or add this unknown word to the (default user) dictionary (**Add to Dictionary**). Adding the unknown word to the dictionary causes the spell-checker to treat the word as correct and to pass on to the next word not found in the dictionary.

After all the words not found in the dictionary have been displayed in turn, and an action taken for each, the spell-checker displays the message: "The spelling check is complete." You can then recheck the document from the beginning (**Recheck Document**) or close the dialog (**Close**).

The **Options...** button opens the Spelling options... dialog, in which you can specify options for the spell check.

8.10.2 Spelling options...

The **Spelling options...** command opens a dialog box (shown below) in which you specify options for the spell check.



Always suggest corrections
Selecting this option causes suggestions from the current dictionary (main dictionary plus listed custom dictionaries) to be displayed in the Suggestions list box. Otherwise no suggestions will be shown.

Make corrections only from main dictionary:
Selecting this option causes only the main dictionary to be used; none of the custom dictionaries is used. Additionally, the Custom Dictionaries... button is disabled, which prevents editing of the custom dictionaries.

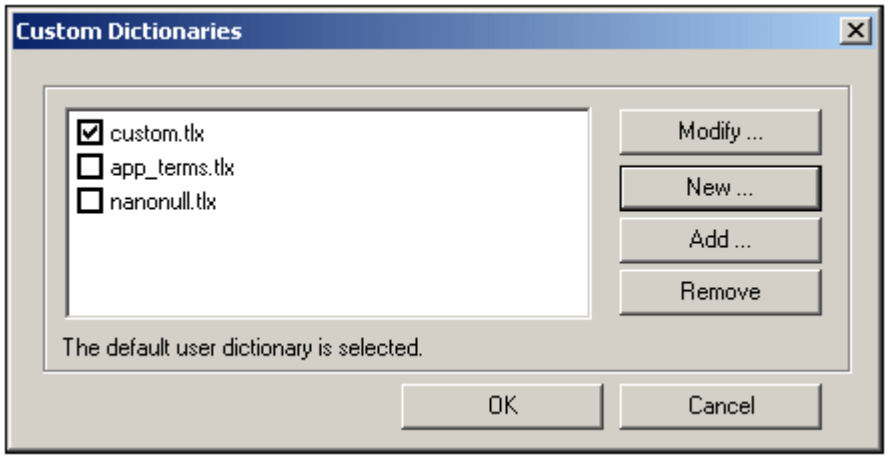
Ignore words in UPPER case:
Selecting this option causes all upper case words to be ignored.

Ignore words with numbers:
Selecting this option causes all words containing numbers to be ignored.

Dictionaries
Each spell-checking round uses the current dictionary. The current dictionary consists of one uneditable main dictionary and the listed custom dictionaries. The number of available main dictionaries is fixed. You select a main dictionary from the drop-down menu in the Dictionary Language combo box. To edit the list of custom dictionaries used in a spell-check, or to edit the contents of a custom dictionary, click the Custom Dictionaries... button and select the required custom dictionary from the list of custom dictionaries.

When you click the **Custom Dictionaries...** button, the following dialog appears:

© 2005 Altova GmbH



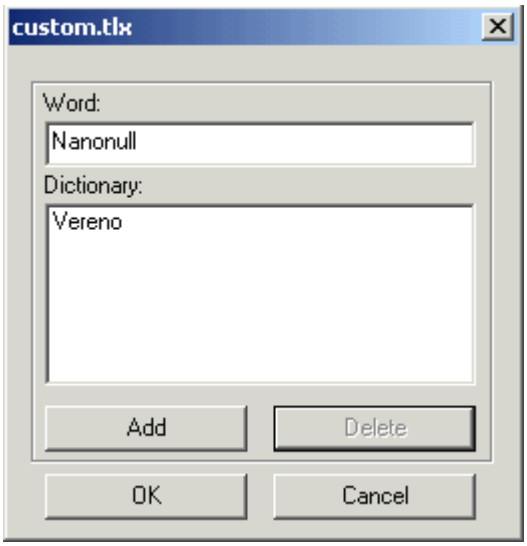
Editing the Custom Dictionaries list
The listed custom dictionaries are part of the current dictionary.

- To add an existing custom dictionary to the list, click the **Add...** button; then browse for the required dictionary, and select it.
- To remove a custom dictionary from the list (and, therefore, from the current dictionary), select the dictionary to be removed and click the **Remove** button. This causes the dictionary to be removed from the list. It is, however, not deleted, and can be added to the list subsequently.
- To create a new custom dictionary and add it to the list, click the **New...** button, open the folder in which the new dictionary is to be created, and give the new dictionary a name. This file must have a `.tlx` suffix.

When you start a spell check, all dictionaries listed in the Custom Dictionaries list box are searched. If you want to limit the search to specific dictionaries, use the Remove command to remove form the list those dictionaries you do not want searched.

The **default user dictionary** is the custom dictionary to which unknown words encountered in a spell-check are added when you click the Add to Dictionary command (during the spell-check). Select the default user dictionary by clicking the check box next to the dictionary you wish to make the default user dictionary.

Modifying the contents of a custom dictionary
To modify the content of a custom dictionary, click the custom dictionary to be modified, and click **Modify...**. This opens the dictionary editor (shown below for the dictionary `custom.tlx`).



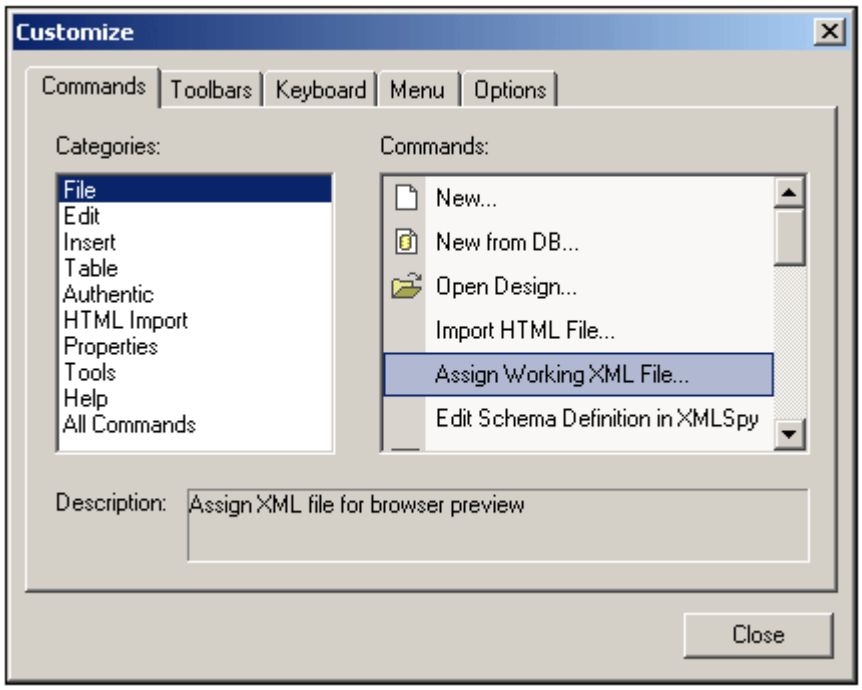
You can now add words to the dictionary and delete words. To add a word, place the cursor in the Word input field, enter the word, and click Add. To delete a word, select the word in the Dictionary pane, and click Delete.

© 2005 Altova GmbH

8.10.3 Customize...

The customize command lets you customize StyleVision to suit your personal needs.

Commands
The **Commands** tab of the Customize dialog allows you to place individual commands in the menu bar and the toolbar.



To add a command to the menu bar or toolbar, select the command in the Commands pane of the Commands tab, and drag it to the menu bar or toolbar. When the cursor is placed over a valid position an I-beam appears, and the command can be dropped at this location. If the location is invalid, a check mark appears. When you drop the command it is created as an icon if the command already has an associated icon; otherwise the command is created as text. After adding a command to the menu bar or toolbar, you can edit its appearance by right-clicking it and then selecting the required action.

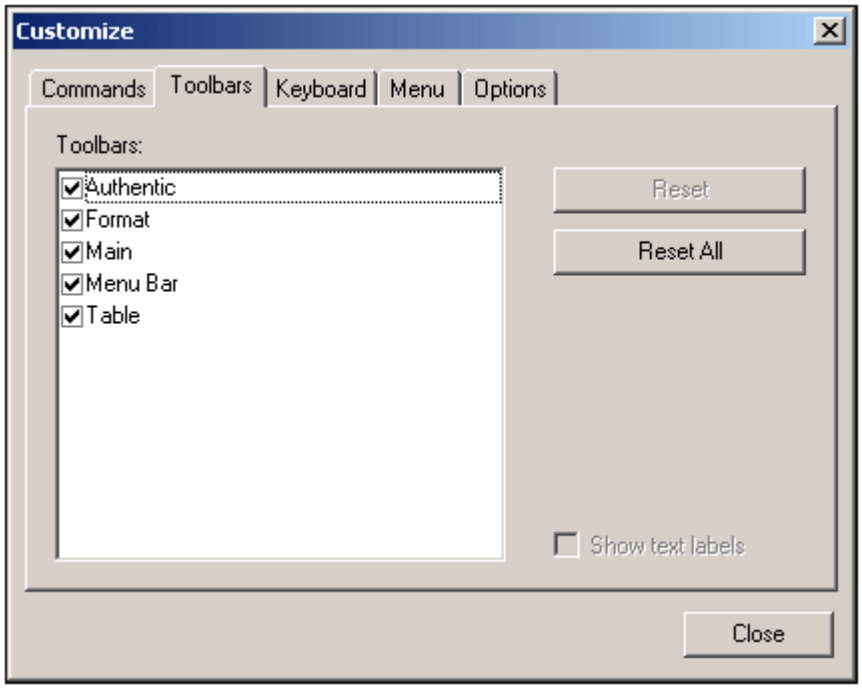
To delete a menu bar or toolbar item, with the Customize dialog open, right-click the item to be deleted, and select Delete.

Note:

- The customization described above applies to the application, and applies whether a document is open in StyleVision or not.
- To reset menus and toolbars to the state they were in when StyleVision was installed, go to the Toolbars tab and click the appropriate Reset button.

See also: Tools.

Toolbars
The **Toolbars** tab allows you to activate or deactivate specific toolbars, to show text labels for toolbar items, and to reset the menu bar and toolbars to their installation state.



The StyleVision interface displays a fixed menu bar and several optional toolbars (Authentic, Format, Main, and Table).

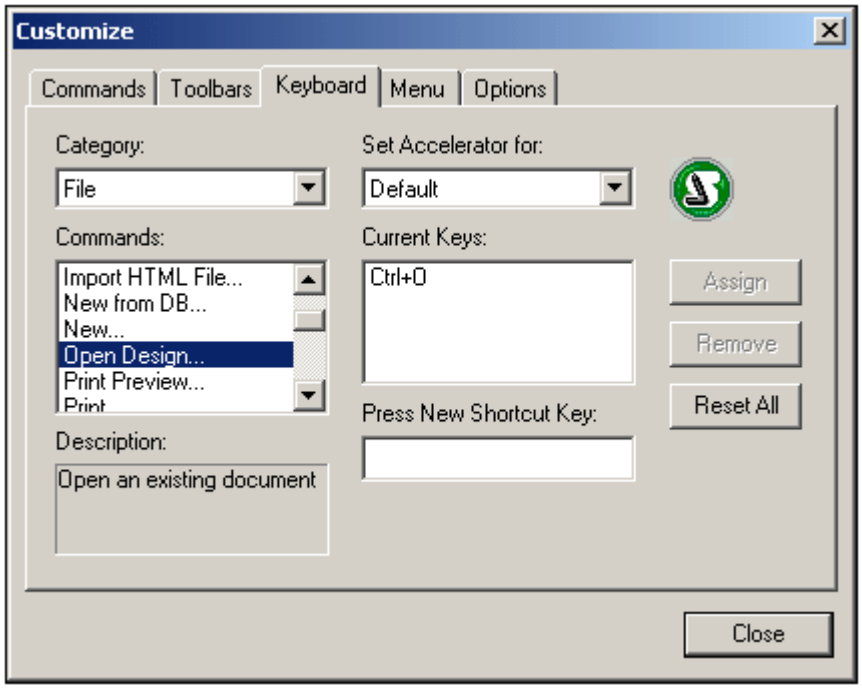
Each toolbar can be divided into groups of commands. Commands can be added to a toolbar via the Commands tab. A toolbar can be dragged from its docked position to any location on the screen. Double-clicking a toolbar's (maximized or minimized) title bar docks and undocks the toolbar.

In the Toolbars tab of the Customize dialog, you can toggle a toolbar on and off by clicking in its checkbox. When a toolbar is selected (in the Toolbars tab), you can cause the text labels of that toolbar's items to be displayed by clicking the **Show text labels** check box. You can also reset a selected toolbar to the state it was in when StyleVision was installed by clicking the **Reset** button. You can reset all toolbars and the menu bar by clicking the **Reset All** button.

Menu Bar
Commands can be added to, and items deleted from, the menu bar: see Commands above. To reset the menu bar to the state it was in when StyleVision was installed, select Menu Bar in the Toolbars tab of the Customize dialog, and click the **Reset** button. (Clicking the **Reset All** button will reset the toolbars as well.)

Keyboard
The **Keyboard** tab allows you to define (or change) keyboard shortcuts for any StyleVision command.

 © 2005 Altova GmbH



To assign a shortcut to a command

1. Select the category in which the command is by using the Category combo box.
2. Select the command you want to assign a shortcut to in the Commands list box.
3. Click in the Press New Shortcut Key input field, and press the shortcut keys that are to activate the command. The shortcut immediately appears in the Press New Shortcut Key input field. If this shortcut has already been assigned to a command, then that command is displayed below the input field. (For example, in the screenshot above, Ctrl+C has already been assigned to the Copy command and cannot be assigned to the Open File command.) To clear the New Shortcut Key input field, press any of the control keys, Ctrl, Alt, or Shift.
4. Click the **Assign** button to permanently assign the shortcut. The shortcut now appears in the Current Keys text box.

To de-assign (or delete) a shortcut

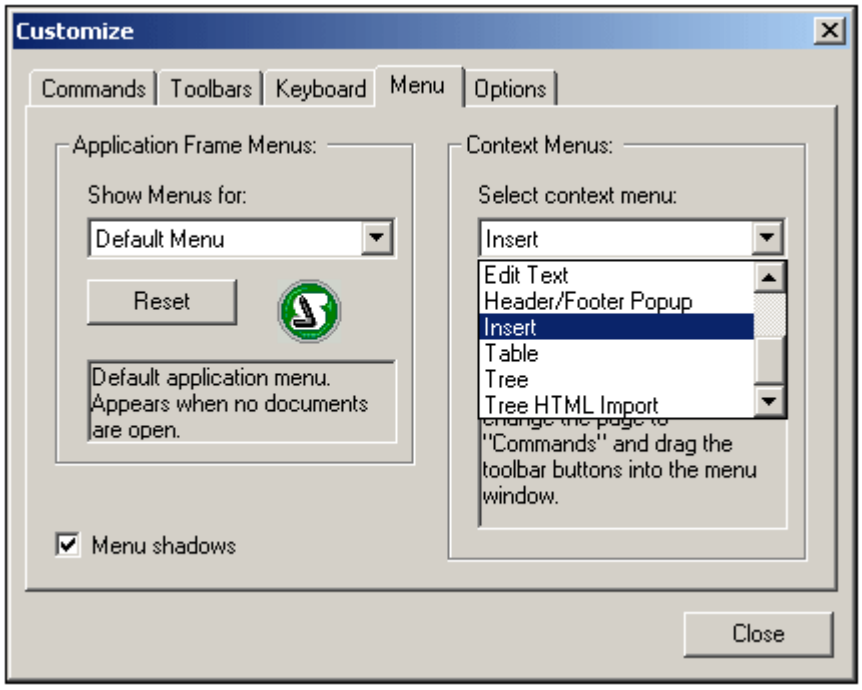
1. Select the command for which the shortcut is to be deleted.
2. Click the shortcut you want to delete in the Current Keys list box.
3. Click the **Remove** button (which has now become active).

To reset all keyboard assignments

1. Click the **Reset All** button to go back to the original, installation-time shortcuts. A dialog box appears prompting you to confirm whether you want to reset all keyboard assignments.
2. Click Yes if you want to reset all keyboard assignments.

Set accelerator for
Currently no function is available.

Menu
The **Menu** tab allows you to customize the main menu bar as well as the context menus (right-click menus).



To customize a menu

1. Select the menu bar you want to customize (Default Menu currently).
2. Click the **Commands** tab, and drag the commands to the menu bar of your choice.

To delete commands from a menu

1. Click right on the command or icon representing the command, and
2. Select the **Delete** option from the popup menu,
   or,
1. Select **Tools | Customize** to open the Customize dialog box, and
2. Drag the command away from the menu and drop it as soon as the check mark icon appears below the mouse pointer.

To reset either of the menu bars

1. Select the Default Menu entry in the combo box)
2. Click the **Reset** button just below the menu name. A prompt appears asking if you are sure you want to reset the menu bar.

To customize a context menu (a right-click menu)

1. Select the context menu from the combo box.
2. Click the **Commands** tab and drag the commands to the context menu that is now open.

To delete commands from a context menu

1. Click right on the command or icon representing the command, and
2. Select the **Delete** option from the popup menu
   or
1. Select **Tools | Customize** to open the Customize dialog box, and
2. Drag the command away from the context menu and drop it as soon as the check mark icon appears below the mouse pointer.

To reset a context menu

1. Select the context menu from the combo box, and
2. Click the **Reset** button just below the context menu name. A prompt appears asking if

© 2005 Altova GmbH

you are sure you want to reset the context menu.

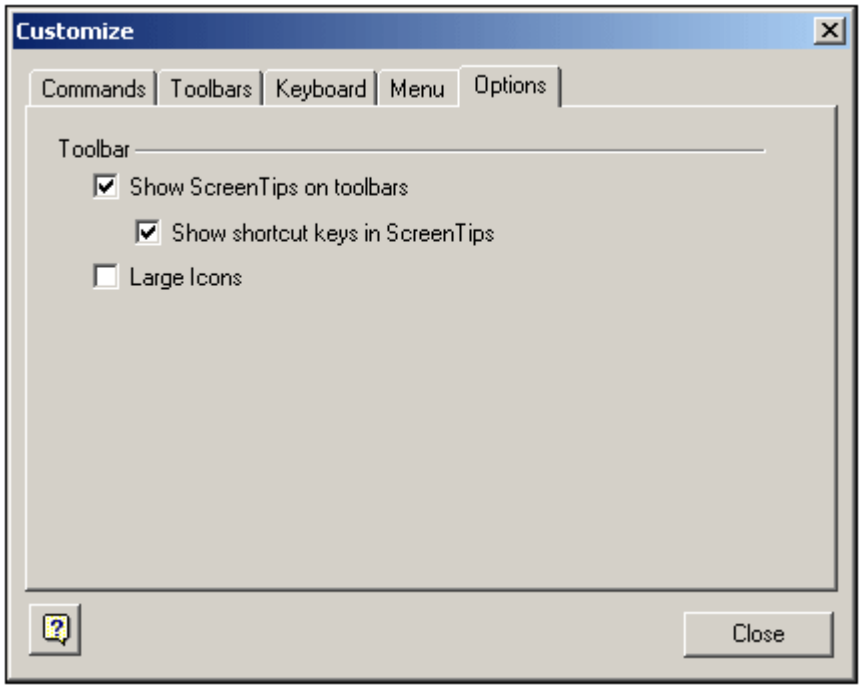
To close a context menu window

- Click on the **Close icon** at the top right of the title bar, or
- Click the Close button of the Customize dialog box.

Menu animations
The menu animation option specifies the way a menu is displayed when a menu is clicked. Select an option from the drop-down list of menu animations.

Menu shadows
If you wish to have menus displayed with a shadow around it, select this option. All menus will then have a shadow.

Options
The **Options** tab allows you to customize additional features of the toolbar.



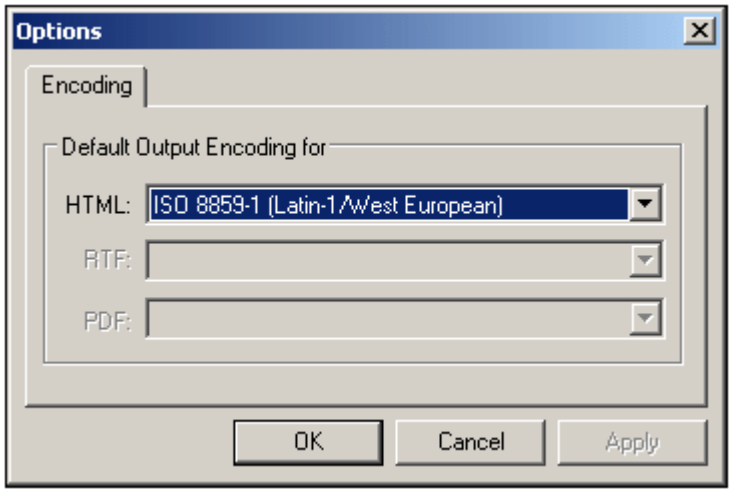
Screen Tips for toolbar items will be displayed if the Show Screen Tips option is checked. The Screen Tips option has a sub-option for whether shortcuts (where available) are displayed in the Screen Tips or not.

Toolbar items can also be displayed as large icons. To do this, check the Large Icons option.

See also: Tools.

8.10.4 Options...

The **Options...** command opens a dialog in which you can specify the encoding of the HTML output file.



To set the encoding of the output HTML file, open the dropdown menu of the combo box and select the desired option from the list of encoding options, and click OK. Every new StyleVision Power Stylesheet you create from this point on, will set the HTML output encoding as defined in this tab.

In the XSLT-for-HTML, the output encoding information is registered at the following locations:

- In the `encoding` attribute of the stylesheet's `xsl:output` element:
  `<xsl:output version="1.0" encoding="UTF-8" indent="no" omit-xml-declaration="no" media-type="text/html" />`
- In the `charset` attribute of the `content-type meta` element in the HTML header:
  `<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />`

Note: This setting is the default encoding for HTML output and will be used for new StyleVision Power Stylesheets. You cannot change the encoding of the currently open StyleVision Power Stylesheet using this dialog. To change the encoding of the currently open StyleVision Power Stylesheet, use the File | Encoding command.

© 2005 Altova GmbH

8.11 Help

The **Help** menu contains commands to access the onscreen help manual for StyleVision, commands to provide information about StyleVision, and links to support pages on the Altova web site. The Help menu also contains the Registration dialog, which lets you enter your license key-code once you have purchased the product.

8.11.1 Table of contents...

The **Table of Contents...** command opens the onscreen help manual for StyleVision with the Table of Contents displayed in the left-hand-side pane of the Help window. The Table of Contents provides a good overview of the entire Help document. Clicking an entry in the Table of Contents takes you to that topic.

© 2005 Altova GmbH

8.11.2 Index...

The **Index...** command opens the onscreen help manual for StyleVision with the Keyword Index displayed in the left-hand-side pane of the Help window. The index lists keywords and lets you navigate to a topic by double-clicking the keyword. If a keyword is linked to more than one topic, you are presented a list of the topics to choose from.

8.11.3 Search...

The **Search...** command opens the onscreen help manual for StyleVision with the Search dialog displayed in the left-hand-side pane of the Help window. To search for a term, enter the term in the input field, and press Return. The Help system performs a full-text search on the entire Help documentation and returns a list of hits. Double-click any item to display it.

© 2005 Altova GmbH

8.11.4 Registration...

When you start StyleVision for the first time, you are presented with the Registration dialog box. You must enter your unique license key-code in this dialog to register your product with Altova. Registering your product is required to activate your license and unlock your software. Eligibility for technical support depends upon the type of license you have purchased, so registration is required in order for you to avail of technical support. If you need to manage your licenses or change licenses, you can access the Registration dialog at any time with the **Registration...** command.

You can register using either a free evaluation key or a permanent key-code. Issues relating to each type of key are described below.

Free evaluation key
To activate the software for a 30-day evaluation, click the **Request FREE evaluation key...** button. Enter your name, company, and e-mail address in the dialog that appears, and click Request Now! The evaluation key is sent to the e-mail address you entered and should reach you in a few minutes. Now enter the key in the key-code field of the registration dialog box and click OK to start working with StyleVision.

When you are ready to order a licensed version of StyleVision, you can use the **Order license key...** button in the registration dialog or **Help | Order form...** to proceed to the secure Altova Online Shop.

License Key-Code
Your permanent license will be of one of two types: a single-user or multi-user license. Both types are sent by e-mail.

A **single-user license** contains your license-data and includes your name, company, e-mail, and key-code. Please make sure that you enter the data required in the registration dialog exactly as given in your license e-mail.

A **multi-user license** contains your license-data and includes your company name and key-code. Please make sure that you enter the data required in the registration dialog exactly as give in your license e-mail. Also enter your personal name in the name field. Note that the StyleVision License Agreement does not allow you to install more than the licensed number of copies of StyleVision on the computers in your organization (per-seat license).

8.11.5 Order form...

The **Order form...** command is a link to the Altova Online Shop. Use this command when you wish to place an order for a licensed version of any Altova software.

© 2005 Altova GmbH

8.11.6 Support Center...

The **Support Center...** command is a link to the Altova Support Center on the Internet. The Support Center provides FAQs, discussion forums where problems are discussed, and access to Altova's technical support staff.

8.11.7 FAQ on the web...

The **FAQ on the web...** command is a link to Altova's FAQ database on the Internet. The FAQ database is constantly updated as Altova support staff encounter new issues raised by customers.

© 2005 Altova GmbH

8.11.8 Components download...

The **Components download...** command is a link to Altova's Component Download Center on the Internet. From here you will be able to download a variety of companion software to use with Altova products. Such software ranges from XSLT and XSL-FO processors to Application Server Platforms. The software available at the Component Download Center is typically free of charge.

8.11.9 StyleVision on the Internet...

The **StyleVision on the Internet...** command is a link to the Altova web site (http://www.altova.com) on the Internet.

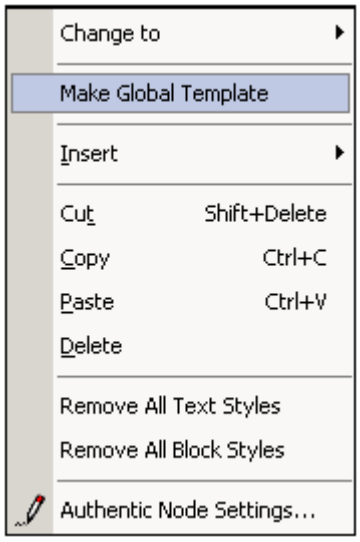
© 2005 Altova GmbH

8.11.10 About StyleVision...

The **About StyleVision...** command displays the splash window and version number of your product.

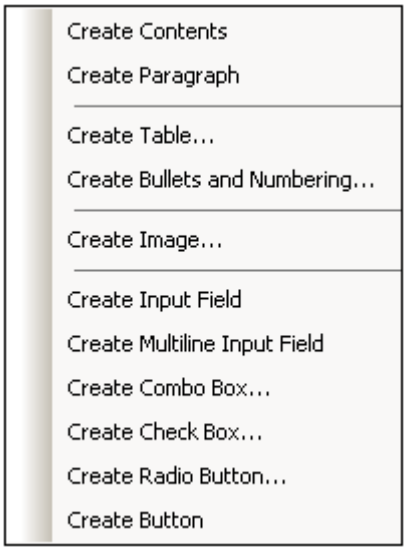
8.12 Context menus

Context menus are opened by right-clicking a StyleVision Power Stylesheet component in the Design Window. What items are displayed in the context menu depends on what component you right-click; they are commands relevant to the selected component. A typical context menu is shown below.



The **Change to** and **Insert** menu items each has a sub-menu. These sub-menus are described in the sub-sections of this section.

There is one other context menu you will encounter. This context menu (shown below) pops up when you drag an element or attribute from the Schema Window and drop it into the Design Window. It requires you to specify what component type the element or attribute is to be created as.

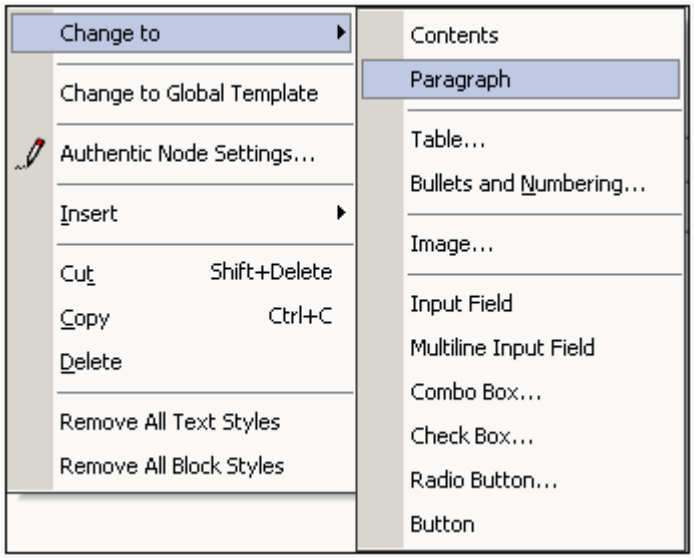


The commands in this menu perform the same task as the corresponding commands in the **Change to** submenu above.

 © 2005 Altova GmbH

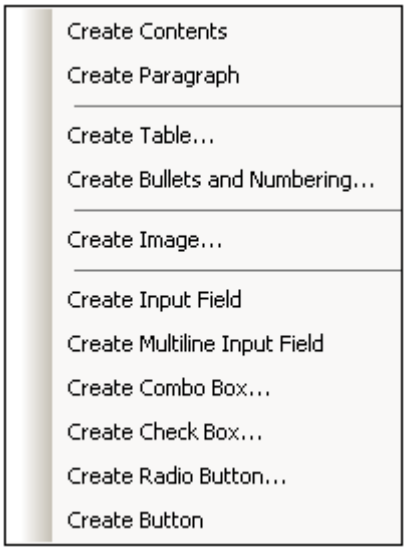
8.12.1 Change to

The **Change to** item of a context menu has a number of subitems. These are shown in the figure below and are described below.



The Change to submenu allows you to change the component type (see Document design process) of the selected node. Note that the Change to submenu is available when a node is selected; it is not available when the component within a node is selected.

The commands in the Change to submenu perform the same tasks as the items in the context menu that appears (shown below) when a element or attribute is dragged from the Schema Window into the Design Window.



The explanations of the Change to commands given below apply to the corresponding Create commands as well.

Change to Contents
The Change to Contents option causes the text content of the selected node (element or attribute) and of all its descendants to be output *in document order*. There will be no whitespace

between text output of descendant elements, and the output document structure and all local inline formatting will disappear. However, if a global template is in use for the selected element or any of its descendants, the inline properties of that template will be retained for that element.

Change to Paragraph
The Change to Paragraph option causes the text content of the selected node (element or attribute) and of all its descendants to be output *in document order* and within a single paragraph. No new paragraph will be generated for any descendant element. There will be no whitespace between descendant elements, and the output document structure and all local inline formatting will disappear. The inline formatting of global templates will, however, be retained.

Change to Table
The Change to Table option changes the component type of the selected element to a dynamic table. It can be applied to elements with children; it cannot be applied to attributes or to elements with no child. How to specify table options is described in Creating dynamic tables.

Change to Bullets and Numbering
The Change to Bullets and Numbering option changes the component type of the selected element to an itemized or a numbered list. A new list item will be created for each instance of the selected element. For usage of lists in StyleVision Power Stylesheets, see Lists.

Change to Image
The Change to Image option changes the component type of the selected node (element or attribute) to an image that you browse for.

Change to [Data-entry device]
Data-entry devices are input fields, combo boxes, check boxes, etc. The various types of data-entry devices used in StyleVision Power Stylesheet are described in the section Data-entry devices. You can change the component type of a selected node to a data-entry device. How to specify settings for the data-entry device is described in Data-entry devices.

© 2005 Altova GmbH

Chapter 9

Appendices

9 Appendices

The appendices contain technical background information and licensing and distribution information relevant to your product.

© 2005 Altova GmbH

9.1 OS and Memory Requirements

Operating System
StyleVision 2005 is a 32-bit Windows application that runs on Windows NT 4.0, Windows 2000, and Windows XP.

Memory
Since StyleVision 2005 is written in C++ it does not require the overhead of a Java Runtime Environment and typically requires less memory than comparable Java-based applications. However, each document is loaded fully into memory so as to parse it completely and to improve viewing and editing speed. The memory requirement increases exponentially with the size of the document. For example, a 512kB document would typically require about 2MB of RAM, whereas a 5MB document can consume up to 50MB. Memory requirements are also influenced by the unlimited Undo history. When repeatedly cutting and pasting large selections in large documents, memory can rapidly be depleted.

9.2 Electronic Software Distribution

StyleVision 2005 is available through electronic software distribution. This provides customers the following unique benefits:

- You can freely evaluate the software before making a purchasing decision.
- Once you decide to buy the software, you can place your order online and immediately get a fully licensed product within minutes.
- You can be sure to always get the latest version of our software.
- The application includes a comprehensive, integrated online help system and an electronic manual that you can also print out if you prefer to read your documentation on paper.

After downloading StyleVision 2005, you can evaluate it for a period of up to 30 days free of charge. During this evaluation period the software will start to remind you after about 20 days that it has not been licensed yet. The reminder message will, however, only be displayed once every time you start the program. If you would like to continue using the program after the 30-day evaluation period, you have to purchase a Software Product License, which is delivered in the form of a key-code that you enter into the Registration dialog to unlock the product. You can register and purchase your license online by directing your browser to access our web shop at http://www.xmlspy.com/order. On this page you will get detailed pricing information (including multi-user discounts) and also find a list of authorized distributors and resellers.

If you want to share StyleVision 2005 with others, please make sure that only the installation program is ever distributed. It contains the application program, grammar description, sample files, and this online manual as well as a quick Read-Me file in one neat package. Any person that receives the StyleVision 2005 package from you is also automatically entitled to a 30-day evaluation period. After the expiration of said period, any other user must also purchase a license in order to be able to use StyleVision 2005.

For further details, please refer to the Software Product License at the end of this manual.

© 2005 Altova GmbH

9.3 License Metering

StyleVision 2005 has a built-in license metering module that helps you avoid any unintentional violation of our license agreement. StyleVision 2005 can be licensed either as a single-user or multi-user product, and the license-metering module ensures that no more than the licensed number of users are using StyleVision 2005 concurrently. This license-metering technology uses your local area network (LAN) to communicate between instances of StyleVision 2005 running on different computers.

Single-user licenses
When StyleVision 2005 starts up, it sends a short broadcast datagram to find any other instance of the product running on another computer in the same network segment. If it doesn't get any response, it will open a port for listening to other instances of StyleVision 2005. Other than that, it will do nothing at all in a single-user situation. If you are not connected to a LAN or are using dial-up connections to connect to the Internet, StyleVision 2005 will NOT generate any network traffic.

Multi-user licenses
If more than one copy of StyleVision 2005 is used within the same LAN, they will briefly communicate with each other on startup to exchange their key-codes to ensure that the number of concurrent licenses purchased is not accidentally violated, as additional copies of the product are launched by more users. This is the same kind of license metering technology, that is common in the Unix world and with many other database development tools. It allows our customers to purchase reasonably-priced concurrent-use multi-user licenses (see http://www.xmlspy.com/order for our price list). Note that StyleVision 2005 at no time attempts to send any information out of your LAN or over the Internet. It also deliberately sends very few and small network packets so as to not put a burden on any network.

The TCP/IP ports (2799) used by StyleVision 2005 are officially registered with the IANA (see http://www.isi.edu/in-notes/iana/assignments/port-numbers for details), and the Altova license-metering module is a proven and tested technology. If you are using a firewall, you may notice communications on port 2799 between the computers that are running StyleVision 2005. You are, of course, free to block such traffic between different groups in your organization, as long as you can ensure by other means, that your license agreement is not violated.

You will also notice that StyleVision 2005 contains many useful functions that make use of your Internet connection, but these are unrelated to the license-metering technology.

9.4 Copyright

All title and copyrights in and to the SOFTWARE PRODUCT (including but not limited to any images, photographs, animations, video, audio, music, text, and "applets" incorporated into the SOFTWARE PRODUCT), the accompanying printed materials, and any copies of the SOFTWARE PRODUCT are owned by Altova GmbH or its suppliers. The SOFTWARE PRODUCT is protected by copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE PRODUCT like any other copyrighted material.

Copyright ©1998-2005 Altova GmbH

All rights reserved.

Patent(s) pending.

The Sentry Spelling-Checker Engine Copyright © 2000 Wintertree Software Inc.

STLport Copyright 1999,2000 Boris Fomitchev, Copyright 1994 Hewlett-Packard Company, Copyright 1996,97 Silicon Graphics Computer Systems, Inc., Copyright 1997 Moscow Center for SPARC Technology.

9.5 Software Product License

THIS IS A LEGAL DOCUMENT -- RETAIN FOR YOUR RECORDS

ALTOVA® END USER LICENSE AGREEMENT

Licensor:

Altova GmbH
Rudolfsplatz 13a/9
A-1010 Wien
Austria

Important - Read Carefully. Notice to User:

This End User License Agreement ("Software License Agreement") is a legal document between you and Altova GmbH ("Altova"). It is important that you read this document before using the Altova-provided software ("Software") and any accompanying documentation, including, without limitation printed materials, 'online' files, or electronic documentation ("Documentation"). By clicking the "I accept" and "Next" buttons below, or by installing, or otherwise using the Software, you agree to be bound by the terms of this Software License Agreement as well as the Altova Privacy Policy ("Privacy Policy") including, without limitation, the warranty disclaimers, limitation of liability, data use and termination provisions below, whether or not you decide to purchase the Software. You agree that this agreement is enforceable like any written agreement negotiated and signed by you. If you do not agree, you are not licensed to use the Software, and you must destroy any downloaded copies of the Software in your possession or control. Please go to our Web site at http://www.altova.com/eula to download and print a copy of this Software License Agreement for your files and http://www.altova.com/privacy to review the privacy policy.

1. SOFTWARE LICENSE

(a) **License Grant.** Upon your acceptance of this Software License Agreement Altova grants you a non-exclusive, non-transferable (except as provided below), limited license to install and use a copy of the Software on your compatible computer, up to the Permitted Number of computers. The Permitted Number of computers shall be delineated at such time as you elect to purchase the Software. During the evaluation period, hereinafter defined, only a single user may install and use the software on one computer. If you have licensed the Software as part of a suite of Altova software products (collectively, the "Suite") and have not installed each product individually, then the Software License Agreement governs your use of all of the software included in the Suite. If you have licensed SchemaAgent, then the terms and conditions of this Software License Agreement apply to your use of the SchemaAgent server software ("SchemaAgent Server") included therein, as applicable and you are licensed to use SchemaAgent Server solely in connection with your use of Altova Software and solely for the purposes described in the accompanying documentation. In addition, if you have licensed XMLSPY Enterprise Edition or MAPFORCE Enterprise Edition, your license to install and use a copy of the Software as provided herein permits you to generate source code based on (i) Altova Library modules that are included in the Software (such generated code hereinafter referred to as the "Restricted Source Code") and (ii) schemas or mappings that you create or provide (such code as may be generated from your schema or mapping source materials hereinafter referred to as the "Unrestricted Source Code"). In addition to the rights granted herein, Altova grants you a non-exclusive, non-transferable, limited license to compile into executable form the complete generated code comprised of the combination of the Restricted Source Code and the Unrestricted Source Code, and to use, copy, distribute or license that executable. You may not distribute or redistribute, sublicense, sell, or transfer to a third party the Restricted Source Code, unless said third party already has a license to the Restricted Source Code through their separate license agreement with Altova or other agreement with Altova.

Altova reserves all other rights in and to the Software.

(b) **Server Use.** You may install one copy of the Software on your computer file server for the purpose of downloading and installing the Software onto other computers within your internal network up to the Permitted Number of computers. If you have licensed .SchemaAgent, then you may install SchemaAgent Server on any server computer or workstation and use it in connection with your Software. No other network use is permitted, including without limitation using the Software either directly or through commands, data or instructions from or to a computer not part of your internal network, for Internet or Web-hosting services or by any user not licensed to use this copy of the Software through a valid license from Altova.

(c) **Concurrent Use**. If you have licensed a "Concurrent-User" version of the Software, you may install the Software on any compatible computers, up to ten (10) times the Permitted Number of users, provided that only the Permitted Number of users actually use the Software at the same time. The Permitted Number of concurrent users shall be delineated at such time as you elect to purchase the Software licenses.

(d) **Backup and Archival Copies.** You may make one backup and one archival copy of the Software, provided your backup and archival copies are not installed or used on any computer and further provided that all such copies shall bear the original and unmodified copyright, patent and other intellectual property markings that appear on or in the Software. You may not transfer the rights to a backup or archival copy unless you transfer all rights in the Software as provided under Section 3.

(e) **Home Use.** You, as the primary user of the computer on which the Software is installed, may also install the Software on one of your home computers for your use. However, the Software may not be used on your home computer at the same time as the Software is being used on the primary computer.

(f) **Key Codes, Upgrades and Updates.** Prior to your purchase and as part of the registration for the thirty (30) -day evaluation period, as applicable, you will receive an evaluation key code. You will receive a purchase key code when you elect to purchase the Software. The purchase key code will enable you to activate the Software beyond the initial evaluation period. You may not re-license, reproduce or distribute any key code except with the express written permission of Altova. If the Software that you have licensed is an upgrade or an update, then the update replaces all or part of the Software previously licensed. The update or upgrade and the associated license keys does not constitute the granting of a second license to the Software in that you may not use the upgrade or update in addition to the Software that it is replacing. You agree that use of the upgrade of update terminates your license to use the Software or portion thereof replaced.

(g) **Title.** Title to the Software is not transferred to you. Ownership of all copies of the Software and of copies made by you is vested in Altova, subject to the rights of use granted to you in this Software License Agreement. As between you and Altova, documents, files, stylesheets, generated program code (including the Unrestricted Source Code) and schemas that are authored or created by you via your utilization of the Software, in accordance with its Documentation and the terms of this Software License Agreement, are your property.

(h) **Reverse Engineering.** Except and to the limited extent as may be otherwise specifically provided by applicable law in the European Union, you may not reverse engineer, decompile, disassemble or otherwise attempt to discover the source code, underlying ideas, underlying user interface techniques or algorithms of the Software by any means whatsoever, directly or indirectly, or disclose any of the foregoing, except to the extent you may be expressly permitted to decompile under applicable law in the European Union, if it is essential to do so in order to achieve operability of the Software with another software program, and you have first requested Altova to provide the information necessary to achieve such operability and Altova has not made such information available. Altova has the right to impose reasonable conditions and to request a reasonable fee before providing such information. Any information supplied by Altova or obtained by you, as permitted hereunder, may only be used by you for the purpose described herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Software. Requests for information from users in the European Union with respect to the above should be directed to the Altova Customer Support Department.

 © 2005 Altova GmbH

(i) **Other Restrictions.** You may not loan, rent, lease, sublicense, distribute or otherwise transfer all or any portion of the Software to third parties except to the limited extent set forth in Section 3 or otherwise expressly provided. You may not copy the Software except as expressly set forth above, and any copies that you are permitted to make pursuant to this Software License Agreement must contain the same copyright, patent and other intellectual property markings that appear on or in the Software. You may not modify, adapt or translate the Software. You may not, directly or indirectly, encumber or suffer to exist any lien or security interest on the Software; knowingly take any action that would cause the Software to be placed in the public domain; or use the Software in any computer environment not specified in this Software License Agreement. You will comply with applicable law and Altova's instructions regarding the use of the Software. You agree to notify your employees and agents who may have access to the Software of the restrictions contained in this Software License Agreement and to ensure their compliance with these restrictions. you agree that you are solely responsible for the accuracy and adequacy of the software for your intended use and you will indemnify and HOLD harmless ALTOVA from any 3rd party suit to the extent based upon the accuracy and adequacy of the software in your use. without limitation, The Software is not intended for use in the operation of nuclear facilities, aircraft navigation, communication systems or air traffic control equipment, where the failure of the Software could lead to death, personal injury or severe physical or environmental damage.

2. INTELLECTUAL PROPERTY RIGHTS

Acknowledgement of Altova's Rights. You acknowledge that the Software and any copies that you are authorized by Altova to make are the intellectual property of and are owned by Altova and its suppliers. The structure, organization and code of the Software are the valuable trade secrets and confidential information of Altova and its suppliers. The Software is protected by copyright, including without limitation by United States Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. You acknowledge that Altova retains the ownership of all patents, copyrights, trade secrets, trademarks and other intellectual property rights pertaining to the Software, and that Altova's ownership rights extend to any images, photographs, animations, videos, audio, music, text and "applets" incorporated into the Software and all accompanying printed materials. You will take no actions which adversely affect Altova's intellectual property rights in the Software. Trademarks shall be used in accordance with accepted trademark practice, including identification of trademark owners' names. Trademarks may only be used to identify printed output produced by the Software, and such use of any trademark does not give you any right of ownership in that trademark. XMLSPY, AUTHENTIC, STYLEVISION, MAPFORCE, MARKUP YOUR MIND, AXAD, NANONULL, and ALTOVA are trademarks of Altova GmbH (registered in numerous countries). Unicode and the Unicode Logo are trademarks of Unicode, Inc. Windows, Windows 95, Windows 98, Windows NT, Windows 2000 and Windows XP are trademarks of Microsoft. W3C, CSS, DOM, MathML, RDF, XHTML, XML and XSL are trademarks (registered in numerous countries) of the World Wide Web Consortium (W3C); marks of the W3C are registered and held by its host institutions, MIT, INRIA and Keio. Except as expressly stated above, this Software License Agreement does not grant you any intellectual property rights in the Software. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web Site.

3. LIMITED TRANSFER RIGHTS

Notwithstanding the foregoing, you may transfer all your rights to use the Software to another person or legal entity provided that: (a) you also transfer each of this Software License Agreement, the Software and all other software or hardware bundled or pre-installed with the Software, including all copies, updates and prior versions, and all copies of font software converted into other formats, to such person or entity; (b) you retain no copies, including backups and copies stored on a computer; (c) the receiving party secures a personalized key code from Altova; and (d) the receiving party accepts the terms and conditions of this Software License Agreement and any other terms and conditions upon which you legally purchased a license to the Software. Notwithstanding the foregoing, you may not transfer education, pre-release, or not-for-resale copies of the Software.

4. PRE-RELEASE AND EVALUATION PRODUCT ADDITIONAL TERMS

If the product you have received with this license is pre-commercial release or beta Software ("Pre-release Software"), then this Section applies. In addition, this section applies to all evaluation and/or demonstration copies of Altova software ("Evaluation Software") and continues in effect until you purchase a license. To the extent that any provision in this section is in conflict with any other term or condition in this Software License Agreement, this section shall supersede such other term(s) and condition(s) with respect to the Pre-release and/or Evaluation Software, but only to the extent necessary to resolve the conflict. You acknowledge that the Pre-release Software is a pre-release version, does not represent final product from Altova, and may contain bugs, errors and other problems that could cause system or other failures and data loss. CONSEQUENTLY, THE PRE-RELEASE AND/OR EVALUATION SOFTWARE IS PROVIDED TO YOU **"AS-IS" WITH NO WARRANTIES FOR USE OR PERFORMANCE,** AND ALTOVA DISCLAIMS ANY WARRANTY OR LIABILITY OBLIGATIONS TO YOU OF ANY KIND, WHETHER EXPRESS OR IMPLIED. WHERE LEGALLY LIABILITY CANNOT BE EXCLUDED FOR PRE-RELEASE AND/OR EVALUATION SOFTWARE, BUT IT MAY BE LIMITED, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF FIFTY DOLLARS (USD $50) IN TOTAL. If the Evaluation Software has a time-out feature, then the software will cease operation after the conclusion of the designated evaluation period. Upon such expiration date, your license will expire unless otherwise extended. Access to any files created with the Evaluation Software is entirely at your risk. You acknowledge that Altova has not promised or guaranteed to you that Pre-release Software will be announced or made available to anyone in the future, that Altova has no express or implied obligation to you to announce or introduce the Pre-release Software, and that Altova may not introduce a product similar to or compatible with the Pre-release Software. Accordingly, you acknowledge that any research or development that you perform regarding the Pre-release Software or any product associated with the Pre-release Software is done entirely at your own risk. During the term of this Software License Agreement, if requested by Altova, you will provide feedback to Altova regarding testing and use of the Pre-release Software, including error or bug reports. If you have been provided the Pre-release Software pursuant to a separate written agreement, your use of the Software is governed by such agreement. You may not sublicense, lease, loan, rent, distribute or otherwise transfer the Pre-release Software. Upon receipt of a later unreleased version of the Pre-release Software or release by Altova of a publicly released commercial version of the Software, whether as a stand-alone product or as part of a larger product, you agree to return or destroy all earlier Pre-release Software received from Altova and to abide by the terms of the license agreement for any such later versions of the Pre-release Software.

5. LIMITED WARRANTY AND LIMITATION OF LIABILITY

(a) **Limited Warranty and Customer Remedies.** Altova warrants to the person or entity that first purchases a license for use of the Software pursuant to the terms of this Software License Agreement that (i) the Software will perform substantially in accordance with any accompanying Documentation for a period of ninety (90) days from the date of receipt, and (ii) any support services provided by Altova shall be substantially as described in Section 6 of this agreement. Some states and jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you. To the extent allowed by applicable law, implied warranties on the Software, if any, are limited to ninety (90) days. Altova's and its suppliers' entire liability and your exclusive remedy shall be, at Altova's option, either (i) return of the price paid, if any, or (ii) repair or replacement of the Software that does not meet Altova's Limited Warranty and which is returned to Altova with a copy of your receipt. This Limited Warranty is void if failure of the Software has resulted from accident, abuse, misapplication, abnormal use, Trojan horse, virus, or any other malicious external code. Any replacement Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. This limited warranty does not apply to Evaluation and/or Pre-release Software

(b) **No Other Warranties and Disclaimer.** THE FOREGOING LIMITED WARRANTY AND REMEDIES STATE THE SOLE AND EXCLUSIVE REMEDIES FOR ALTOVA OR

 © 2005 Altova GmbH

ITS SUPPLIER'S BREACH OF WARRANTY. ALTOVA AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, AND FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, ALTOVA AND ITS SUPPLIERS MAKE NO WARRANTIES, CONDITIONS, REPRESENTATIONS OR TERMS, EXPRESS OR IMPLIED, WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE AS TO ANY OTHER MATTERS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, ALTOVA AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, INFORMATIONAL CONTENT OR ACCURACY, QUIET ENJOYMENT, TITLE AND NON-INFRINGEMENT, WITH REGARD TO THE SOFTWARE, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE/JURISDICTION TO STATE/JURISDICTION.

(c) **Limitation Of Liability.** TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW EVEN IF A REMEDY FAILS ITS ESSENTIAL PURPOSE, IN NO EVENT SHALL ALTOVA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF ALTOVA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, ALTOVA'S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS SOFTWARE LICENSE AGREEMENT SHALL BE LIMITED TO THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE PRODUCT. Because some states and jurisdictions do not allow the exclusion or limitation of liability, the above limitation may not apply to you. In such states and jurisdictions, Altova's liability shall be limited to the greatest extent permitted by law and the limitations or exclusions of warranties and liability contained herein do not prejudice applicable statutory consumer rights of person acquiring goods otherwise than in the course of business. The disclaimer and limited liability above are fundamental to this Software License Agreement between Altova and you.

(d) **Infringement Claims.** Altova will indemnify and hold you harmless and will defend or settle any claim, suit or proceeding brought against you by a third party that is based upon a claim that the content contained in the Software infringes a copyright or violates an intellectual or proprietary right protected by United States or European Union law ("Claim"), but only to the extent the Claim arises directly out of the use of the Software and subject to the limitations set forth in Section 5 of this Agreement except as otherwise expressly provided. You must notify Altova in writing of any Claim within ten (10) business days after you first receive notice of the Claim, and you shall provide to Altova at no cost with such assistance and cooperation as Altova may reasonably request from time to time in connection with the defense of the Claim. Altova shall have sole control over any Claim (including, without limitation, the selection of counsel and the right to settle on your behalf on any terms Altova deems desirable in the sole exercise of its discretion). You may, at your sole cost, retain separate counsel and participate in the defense or settlement negotiations. Altova shall pay actual damages, costs, and attorney fees awarded against you (or payable by you pursuant to a settlement agreement) in connection with a Claim to the extent such direct damages and costs are not reimbursed to you by insurance or a third party, to an aggregate maximum equal to the purchase price of the Software. If the Software or its use becomes the subject of a Claim or its use is enjoined, or if in the opinion of Altova's legal counsel the Software is likely to become the subject of a Claim, Altova shall attempt to resolve the Claim by using commercially reasonable efforts to modify the Software or obtain a license to continue using the Software. If in the opinion of Altova's legal counsel the Claim, the injunction

or potential Claim cannot be resolved through reasonable modification or licensing, Altova, at its own election, may terminate this Software License Agreement without penalty, and will refund to you on a pro rata basis any fees paid in advance by you to Altova. THE FOREGOING CONSTITUTES ALTOVA'S SOLE AND EXCLUSIVE LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT. This indemnity does not apply to infringements that would not be such, except for customer-supplied elements.

6. SUPPORT AND MAINTENANCE

Altova offers multiple optional "Support & Maintenance Package(s)" ("SMP") for the version of Software product edition that you have licensed, which you may elect to purchase in addition to your Software license. The Support Period, hereinafter defined, covered by such SMP shall be delineated at such time as you elect to purchase a SMP. Your rights with respect to support and maintenance as well as your upgrade eligibility depend on your decision to purchase SMP and the level of SMP that you have purchased:

(a) If you have not purchased SMP, you will receive the Software AS IS and will not receive any maintenance releases or updates. However, Altova, at its option and in its sole discretion on a case by case basis, may decide to offer maintenance releases to you as a courtesy, but these maintenance releases will not include any new features in excess of the feature set at the time of your purchase of the Software. In addition, Altova will provide free technical support to you for thirty (30) days after the date of your purchase (the "Support Period" for the purposes of this paragraph a), and Altova, in its sole discretion on a case by case basis, may also provide free courtesy technical support during your thirty (30)-day evaluation period. Technical support is provided via a Web-based support form only, and there is no guaranteed response time.

(b) If you have purchased SMP, then solely for the duration of its delineated Support Period, **you are eligible to receive the version of the Software edition** that you have licensed and all maintenance releases and updates for that edition that are released during your Support Period. For the duration of your SMP's Support Period, you will also be eligible to receive upgrades to the comparable edition of the next version of the Software that succeeds the Software edition that you have licensed for applicable upgrades released during your Support Period. The specific upgrade edition that you are eligible to receive based on your Support Period is further detailed in the SMP that you have purchased. Software that is introduced as separate product is not included in SMP. Maintenance releases, updates and upgrades may or may not include additional features. In addition, Altova will provide Priority Technical Support to you for the duration of the Support Period. Priority Technical Support is provided via a Web-based support form only, and Altova will make commercially reasonable efforts to respond via e-mail to all requests within forty-eight (48) hours during Altova's business hours (MO-FR, 8am UTC – 10pm UTC, Austrian and US holidays excluded) and to make reasonable efforts to provide work-arounds to errors reported in the Software.

During the Support Period you may also report any Software problem or error to Altova. If Altova determines that a reported reproducible material error in the Software exists and significantly impairs the usability and utility of the Software, Altova agrees to use reasonable commercial efforts to correct or provide a usable work-around solution in an upcoming maintenance release or update, which is made available at certain times at Altova's sole discretion.

If Altova, in its discretion, requests written verification of an error or malfunction discovered by you or requests supporting example files that exhibit the Software problem, you shall promptly provide such verification or files, by email, telecopy, or overnight mail, setting forth in reasonable detail the respects in which the Software fails to perform. You shall use reasonable efforts to cooperate in diagnosis or study of errors. Altova may include error corrections in maintenance releases, updates, or new major releases of the Software. Altova is not obligated to fix errors that are immaterial. Immaterial errors are those that do not significantly impact use of the Software. Whether or not you have purchased the Support & Maintenance Package, technical support only covers issues or questions resulting directly out of the operation of the Software and Altova will not provide you with generic consultation, assistance, or advice under any circumstances.

Updating Software may require the updating of software not covered by this Software License Agreement before installation. Updates of the operating system and application software not specifically covered by this Software License Agreement are your responsibility and will not be provided by Altova under this Software License Agreement. Altova's obligations under this Section 6 are contingent upon your proper use of the Software and your compliance with the terms and conditions of this Software License Agreement at all times. Altova shall be under no obligation to provide the above technical support if, in Altova's opinion, the Software has failed due to the following conditions: (i) damage caused by the relocation of the software to another location or CPU; (ii) alterations, modifications or attempts to change the Software without Altova's written approval; (iii) causes external to the Software, such as natural disasters, the failure or fluctuation of electrical power, or computer equipment failure; (iv) your failure to maintain the Software at Altova's specified release level; or (v) use of the Software with other software without Altova's prior written approval. It will be your sole responsibility to: (i) comply with all Altova-specified operating and troubleshooting procedures and then notify Altova immediately of Software malfunction and provide Altova with complete information thereof; (ii) provide for the security of your confidential information; (iii) establish and maintain backup systems and procedures necessary to reconstruct lost or altered files, data or programs.

7. SOFTWARE ACTIVATION, UPDATES AND LICENSE METERING

(a) **License Metering**. Altova has a built-in license metering module that helps you to avoid any unintentional violation of this Software License Agreement. Altova may use your internal network for license metering between installed versions of the Software.

(b) **Software Activation**. **Altova's Software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of installation, registration or update to an Altova-operated license server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the Software and to improve customer service. Activation is based on the exchange of license related data between your computer and the Altova license server. You agree that Altova may use these measures and you agree to follow any applicable requirements.**

(c) **LiveUpdate**. Altova provides a new LiveUpdate notification service to you, which is free of charge. Altova may use your internal network and Internet connection for the purpose of transmitting license-related data to an Altova-operated LiveUpdate server to validate your license at appropriate intervals and determine if there is any update available for you.

(d) **Use of Data.** The terms and conditions of the Privacy Policy are set out in full at http://www.altova.com/privacy and are incorporated by reference into this Software License Agreement. By your acceptance of the terms of this Software License Agreement or use of the Software, you authorize the collection, use and disclosure of information collected by Altova for the purposes provided for in this Software License Agreement and/or the Privacy Policy as revised from time to time. European users understand and consent to the processing of personal information in the United States for the purposes described herein. Altova has the right in its sole discretion to amend this provision of the Software License Agreement and/or Privacy Policy at any time. You are encouraged to review the terms of the Privacy Policy as posted on the Altova Web site from time to time.

8. TERM AND TERMINATION

This Software License Agreement may be terminated (a) by your giving Altova written notice of termination; or (b) by Altova, at its option, giving you written notice of termination if you commit a breach of this Software License Agreement and fail to cure such breach within ten (10) days after notice from Altova. In addition the Software License Agreement governing your use of a previous version that you have upgraded or updated of the Software is terminated upon your acceptance of the terms and conditions of the Software License Agreement accompanying such upgrade or update. Upon any termination of the Software License Agreement, you must cease all use of the Software that it governs, destroy all copies then in your possession or control and take such other actions as Altova may reasonably request to ensure that no copies of the Software remain in your possession or control. The terms and conditions set forth in Sections 1(g), (h), (i),

2, 5(b), (c), 9, and 10 survive termination as applicable

9. RESTRICTED RIGHTS NOTICE AND EXPORT RESTRICTIONS.

The Software was developed entirely at private expense and is commercial computer software provided with **RESTRICTED RIGHTS**. Use, duplication or disclosure by the U.S. Government or a U.S. Government contractor or subcontractor is subject to the restrictions set forth in this Agreement and as provided in FAR 12.211 and 12.212 (48 C.F.R. §12.211 and 12.212) or DFARS 227. 7202 (48 C.F.R. §227-7202) as applicable. Consistent with the above as applicable, Commercial Computer Software and Commercial Computer Documentation licensed to U.S. government end users only as commercial items and only with those rights as are granted to all other end users under the terms and conditions set forth in this Software License Agreement. Manufacturer is Altova GmbH, Rudolfsplatz, 13a/9, A-1010 Vienna, Austria/EU. You may not use or otherwise export or re-export the Software or Documentation except as authorized by United States law and the laws of the jurisdiction in which the Software was obtained. In particular, but without limitation, the Software or Documentation may not be exported or re-exported (i) into (or to a national or resident of) any U.S. embargoed country or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce's Table of Denial Orders. By using the Software, you represent and warrant that you are not located in, under control of, or a national or resident of any such country or on any such list.

10. GENERAL PROVISIONS

If you are located in the European Union and are using the Software in the European Union and not in the United States, then this Software License Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht, Wien (Commercial Court, Vienna) in connection with any such dispute or claim.

If you are located in the United States or are using the Software in the United States then this Software License Agreement will be governed by and construed in accordance with the laws of the Commonwealth of Massachusetts, USA (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the federal or state courts of Massachusetts and you further agree and expressly consent to the exercise of personal jurisdiction in the federal or state courts of Massachusetts in connection with any such dispute or claim.

If you are located outside of the European Union or the United States and are not using the Software in the United States, then this Software License Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht Wien (Commercial Court, Vienna) in connection with any such dispute or claim. This Software License Agreement will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

This Software License Agreement contains the entire agreement and understanding of the parties with respect to the subject matter hereof, and supersedes all prior written and oral understandings of the parties with respect to the subject matter hereof. Any notice or other communication given under this Software License Agreement shall be in writing and shall have been properly given by either of us to the other if sent by certified or registered mail, return receipt requested, or by overnight courier to the address shown on Altova's Web site for Altova and the address shown in

 © 2005 Altova GmbH

Altova's records for you, or such other address as the parties may designate by notice given in the manner set forth above. This Software License Agreement will bind and inure to the benefit of the parties and our respective heirs, personal and legal representatives, affiliates, successors and permitted assigns. The failure of either of us at any time to require performance of any provision hereof shall in no manner affect such party's right at a later time to enforce the same or any other term of this Software License Agreement. This Software License Agreement may be amended only by a document in writing signed by both of us. In the event of a breach or threatened breach of this Software License Agreement by either party, the other shall have all applicable equitable as well as legal remedies. Each party is duly authorized and empowered to enter into and perform this Software License Agreement. If, for any reason, any provision of this Software License Agreement is held invalid or otherwise unenforceable, such invalidity or unenforceability shall not affect the remainder of this Software License Agreement, and this Software License Agreement shall continue in full force and effect to the fullest extent allowed by law. The parties knowingly and expressly consent to the foregoing terms and conditions.

Last updated: 2005-03-04

Index

A

Add element/attribute prompt, 275
for Authentic View, 275
Add to parent, 291
HTML import, 291
Additional Validation, 79
in Authentic View, 79
setting in StyleVision Power Stylesheets, 79
settings, 275
Aligning table cell content, 273
in SPSs, 273
Altova web site, 322
Anchors, 56
in SPS, 56
Append, 293
HTML import, 293
to table in SPS, 260
Append row, 259
to table in SPS, 259
Appendices, 328
Apply, 211
formatting, 211
Assign Working XML File, 222
to SPS, 222
Attribute values, 36
in SPS, 36
attributeFormDefault, 89, 91, 94
Attributes, 204, 206
creating HTML content as ,in HTML to XML, 204, 206
in HTML to XML conversion, 197, 207
Authentic, 33
dynamic table editing, 152
in StyleVision, 274
markup display, 152
Authentic node settings, 79
additional validation, 79
dialog box, 81
in SPS, 275
Authentic Preview, 227
save edits, 227
Authentic View, 152
changing view to, 152
context menus, 156
document display, 154
editing dynamic tables, 152
entry helpers in, 151
formatting text in, 152
in Altova products, 4
inserting tables, 280
interface, 150
main window, 154
main window in, 151
markup display in, 152, 154
menu bar in, 151
of Working XML file, 32
overview of GUI, 151
project window in, 151
SPS Tables, 158
status bar in, 151
tables (SPS and XML), 158
toolbar icons, 152
toolbar in, 151
usage of important features, 157
usage of XML tables, 159
XML table icons, 162
XML tables, 159
XPath to selected node, 151
Auto-add Date Picker, 284
Auto-Calculation, 59
in SPS, 59
in SPSs, 253
how to use, 130

B

Background color, 59
in Authentic tables, 59
Base year, 138
in input formatting, 138
bgcolor, 59
in Authentic tables, 59
Block components, 44
in SPS, 44
Block formatting, 21
in SPS, 21
Block Style Window, 18

© 2005 Altova GmbH

Block Style Window, 18
for formatting in SPS, 44
Block styles, 215
in SPSs, 215
Bookmarks, 56
in SPS, 56
in SPSs, 251
Borders, 271
of SPS tables, 271
Bullets and numbering, 298
changing styles in SPS, 298
in SPSs, 245
Bulletted lists, 245
in SPSs, 245
Button, 106
creating in Authentic Stylesheets, 106
in StyleVision Power Stylesheets, 112

C

CALS tables, 280
in SPSs, 280
in StyleVision Power Stylesheets, 126
Carriage return, 169
see Enter key, 169
Cell, 269
split horizontally, 269
split vertically, 270
in StyleVision Power Stylesheet tables, 124
Cellpadding, 272
in SPS tables, 272
Cells, 265, 266, 267, 268
joining in SPS tables, 265, 266, 267, 268
Change to, 325
submenu in StyleVision, 325
Changing schema, 173
effect on StyleVision Power Stylesheet, 173
effect on SPS, 173
Changing view, 152
to Authentic View, 152
Check box, 106
creating in Authentic Stylesheet, 106
creating in StyleVision Power Stylesheet, 111
Child nodes to add, 275
in Authentic View, 275
Close SPS, 220
Column, 260
append to table in SPS, 260
delete from table in SPS, 264
insert in table in SPS, 262
Combined schemas, 94
and namespace prefixing, 94
Combo box, 106
creating dropdown list items with XPath, 106
creating in Authentic Stylesheet, 109
getting values from XML file, 109
Command line, 145
and parameters, 145
using StyleVision from, 148
Companion software, 321
for download, 321
Component download center, 321
at Altova web site, 321
Component types, 325
changing in SPS, 325
creating, 325
Components, 40
moving in SPS, 40
moving, 174
Conditional template, 176
for processing siblings, 176
for expanding/collapsing elements, 133
how to use, 133
in SPS, 64
in SPSs, 255
in StyleVision, 255
moving in StyleVision Power Stylesheets, 174
Conditions, 64
in SPS, 64
Content model, 103
correct design, 103
Contents, 247
in SPSs, 247
Context menus, 156
in Authentic View, 156
in StyleVision, 324
Convert, 289
HTML to XML, 289
Copy command, 233
in StyleVision, 233
Create SPS, 218
from schema, 218
Custom dictionaries, 304
for SPS spell-checks, 304

© 2005 Altova GmbH

Customizing, 307
StyleVision, 307
Cut, 211
and copy in Designer, 211
in StyleVision, 233

D

Databases, 3
in StyleVision (Enterprise Edition), 3
Data-entry device, 301
properties of, 301
how to use, 106
in SPS, 36
Date, 138
formatting of, 138
creating the Date Picker, 136
Date Picker, 284
auto-add, 284
how to use, 136
insert, 249
using in Authentic View, 165
DB, 275
maximum records to display, 275
parameters in DB Filters, 145
creating and editing, 239
Decimals, 138
formatting of, 138
Default user dictionary, 304
for SPS spell-checks, 304
Delete, 258
table in SPS, 258
from table in SPS, 264
Delete command, 233
in StyleVision, 233
Delete row, 263
from table in SPS, 263
Deleting schema nodes, 206
in HTML to XML conversion, 206
Deleting schema tree nodes, 188
in HTML to XML conversion, 188
Design Document, 174
moving components, 174
selecting nodes for placement, 174
Design Window, 211
symbols and icons, 211
Dictionaries, 304
for SPS spell-checks, 304
Document Design, 22
general description, 22
Document element, 7
definition, 7
in HTML to XML conversion, 186
Document node, 7
definition, 7
Document root, 7
definition, 7
Document structure, 8
overview, 8
Documentation, 14
about, 14
DTDs and namespaces, 88
Dynamic (SPS) tables in Authentic View, 158
usage of, 158
Dynamic components/content, 28
Dynamic images, 38
in SPS, 38
in SPSs, 242
Dynamic tables, 121
and global templates, 121
editing, 152
how to create, 121
how to format, 124
in SPS, 48
in StyleVision Power Stylesheets, 117
modifying properties of ,in SPS, 48

E

Edit menu, 233
in StyleVision, 233
Editing XSLT, 177
manually, 177
Element, 33
inserting in SPS, 33
and global templates, 96
Element creation, 188, 200
in HTML to XML conversion, 188, 200
elementFormDefault, 89, 91, 94
Elements, 103
correct design for addinng, 103
creating HTML content as ,in HTML to XML, 204, 206

© 2005 Altova GmbH

Elements, 103
inserting in SPS, 40
Encoding, 312
for output files, 312
changing in active SPS, 224
Enter key, 169
effects of using, 169
Entities, 166, 287
defining in Authentic View, 166, 287
unparsed, 147
Evaluation key, 317
for your Altova software, 317
Exit, 232
StyleVision, 232
Expanding and collapsing elements in Authentic View, 133

F

FAQs on StyleVision, 320
File, 220
closing SPS, 220
opening SPS, 219
File menu, 217
Files, 231
open recently used in StyleVision, 231
Find, 236
command in Authentic Preview, 236
command in Authentic Preview, 238
FO, 17
compliance levels (Enterprise Edition), 17
setting up in StyleVision Enterprise Edition, 17
Footers, 124
for large tables, 124
in Authentic tables, 73
in SPSs, 256
FOP, 17
in StyleVision Enterprise Edition, 17
Format, 211
marking, 211
defining for Inout Formatting, 299
Formatting, 44
in SPS, 44
in SPSs, 215, 244
in StyleVision, 21
of numeric fields, 138
of tables in SPSs, 272
text in Authentic View, 278
in Authentic Stylesheets, 215
Formatting text, 77
in SPS, 77

G

Generated files, 30
saving, 30
Generated XSLT, 177
editing manually, 177
Generating output, 201
in HTML to XML conversion, 201
Global templates, 96
and dynamic tables, 96
and mixed content, 96
creating and using, 96
description of, 8
effect on rest-of-contents, 102
for Authentic text formatting, 77
in HTML to XML conversion, 207
in interface, 20
in SPS, 77
Graphics formats, 168
in Authentic View, 168

H

Headers, 124
for large PDF tables, 124
in SPSs, 256
difference from appended/inserted rows, 121
Help menu, 313, 314, 315, 316
in StyleVision, 313, 314, 315, 316
Hide markup, 152, 154
Horizontal line, 246
inserting in an SPS, 246
Horizontal rule, 301
properties of, 301
HTML, 291
add to parent, 291
and SPSs, 6
append, 293

© 2005 Altova GmbH

HTML, 291
insert, 292
open file, 290
settings, 294
import into StyleVision, 221
importing for conversion to XML, 186, 203
HTML import, 289
HTML output, 24
code of XSLT stylesheet for, 24
specifying properties for in SPS, 296
HTML Preview, 25
general description, 25
HTML tables, 280
in SPSs, 280
in StyleVision Power Stylesheets, 126
HTML to XML, 207
attributes, 207
converting HTML hyperlinks, 199
converting HTML images, 198
converting HTML lists, 192
converting HTML tables, 195
creating element nodes, 188, 200
creating HTML content as nodes, 204, 206
creating schema tree attributes, 197
delete node, 188
generating output, 201
global templates, 207
HTML file import, 186
moving nodes in the schema tree, 197
multiple node instances, 200
open HTML file, 203
points to note, 207
quick reference, 202
renaming nodes in schema tree, 188
Root Template, 207
same name siblings, 200
saving schema, 187
saving the document, 207
saving XML file, 187
undo command, 188, 207
viewing generated schema, 187
viewing generated XML, 187
HTML to XML conversion, 184
tutorial, 185
Hyperlinks, 199
in HTML to XML conversion, 199
in SPS, 56
in SPSs, 252
Hyphenation, 180
in PDF output, 180

I

Icons, 77
for Authentic View text, 77
for formatting text in Authentic View, 278
in SPS, 77
in StyleVision, 211
Image formats, 168
in Authentic View, 168
Images, 198
in HTML to XML conversion, 198
in SPS, 38
in SPSs, 242
Import, 289
HTML, 289
and namespaces, 89
Importeds chemas, 94
and namespace prefixing, 94
Importing HTML, 186, 203
for conversion to XML, 186, 203
Input field, 106
creating in <AUTH-STYLE>s, 106
how to use, 108
in SPS, 36
Input formatting, 299
defining format strings, 299
Insert, 292
HTML import, 292
in table in SPS, 262
Insert menu, 241
StyleVision, 241
Insert row, 261
in table in SPS, 261
Integer, 138
formatting of, 138
Interface, 18
description of, 18

© 2005 Altova GmbH

J

Joining cells, 265, 266, 267, 268
in SPS tables, 265, 266, 267, 268

K

Key-codes, 317
for your Altova software, 317

L

License metering, 331
Licenses, 317
for your Altova software, 317
Links, 199
in HTML to XML conversion, 199
in SPS, 56
specifying colors of in SPS, 296
List properties, 298
in SPS, 298
Lists, 113
creating in StyleVision Power Stylesheets, 113
in HTML to XML conversion, 192
in SPSs, 245
moving in StyleVision Power Stylesheets, 174
Local templates, 8
description of, 8

M

Manually editing XSLT, 177
Marking, 211
table data, 211
Markup, 152, 154, 286
in Authentic View, 152, 154, 286
Menu Bar, 26
general description, 26
Mixed content, 96
and global templates, 96
Mixed markup mode, 275
settings, 275
Modifying schema, 173
effect on StyleVision Power Stylesheet, 173
effect on SPS, 173
Modifying table properties, 48
in SPS, 52, 59
Modifyng table properties, 73
in SPS, 73
Moving components, 40
in SPS, 40
Moving schema tree nodes, 197
in HTML to XML conversion, 197
Multiline input field, 106
creating in <AUTH-STYLE>s, 106
Multiple node instances, 200
in HTML to XML conversion, 200

N

Namespace prefixes in <AUTH-STYLE>s, 91, 94
Namespaces, 89
and imported schemas, 89
and namespace prefixes, 89
and qualified elements and attributes, 89
and schemas with a single namespace, 91
and schemas with multiple namespaces, 94
in <AUTH-STYLE>s, 88, 91, 94
in DTDs and XML Schemas, 88
Nested dynamic tables, 121
Nested lists, 113
creating in StyleVision Power Stylesheets, 113
Node settings, 275
in SPS, 275
Nodes, 204, 206
creating HTML content as ,in HTML to XML, 204, 206
selecting in StyleVision, 174
Numbered lists, 245
in SPSs, 245
Numbers, 138
formatting of, 138
Numeric fields, 138
formatting of, 138

© 2005 Altova GmbH

O

Open, 290
HTML file, 290
recently used StyleVision files, 231
for conversion to XML, 186, 203
Open SPS, 219
Ordering Altova software, 318
Output encoding, 224
changing in active SPS, 224
Output files, 148
using command line to generate, 148

P

Page breaks, 250
in SPSs, 250
Page numbers, 250
in SPSs, 250
Paragraph formats, 243
in SPS, 243
Parameters, 239
creating and editing, 239
how to use, 145
Paste, 211
cut and copy in Designer, 211
in StyleVision, 233
PDF in Enterprise Edition, 6
design considerations, 6
PDF Preview, 17
setting up in StyleVision, 17
Points to note, 207
HTML to XML conversion, 207
position() function, 176
selecting with, 176
Predefined format strings, 299
for input formatting, 299
Pre-defined formats, 44
in SPS, 44
in SPSs, 215, 244
removing, 215
Predicates, 176
in StyleVision Power Stylesheets, 176
Preview of Working XML file, 33
in StyleVision, 33
Print command, 230
Print preview, 229
of Working XML File, 229
Privacy, 331
Processors, 321
for download, 321
Properties, 295
menu for SPS, 295
table, 297

Q

Qualifying elements and attributes, 89, 91, 94
Quick reference, 202
HTML to XML conversion, 202

R

Radio button, 106
creating in Authentic Stylesheets, 106
in StyleVision Power Stylesheets, 112
Recently used files, 231
in StyleVision, 231
Redo command, 235
in StyleVision, 235
Reference, 202
HTML to XML conversion, 202
Registering your Altova software, 317
Regular expressions, 236
in Find command, 236
Removing pre-defined formats, 215
Renaming schema nodes, 206
in HTML to XML conversion, 206
Renaming schema tree nodes, 188
in HTML to XML conversion, 188
Rest of Contents, 248
in SPSs, 248
Rest-of-contents, 102
description of, 102
Return key, 215
and pre-defined formats, 215
see Enter key, 169

© 2005 Altova GmbH

Root element, 186
in HTML to XML conversion, 186
Root node, 7
definition, 7
Root template, 8
description of, 8
in HTML to XML conversion, 186, 207
is empty in schema tree, 173
Row, 259
append to table in SPS, 259
delete from table in SPS, 263
insert in table in SPS, 261
RTF in Enterprise Edition, 6
design considerations, 6
Rule, 246
inserting in an SPS, 246

S

Same name siblings, 200
in HTML to XML conversion, 200
Save, 228
files generated from StyleVision, 228
Working XML File, 227
in StyleVision, 226
Save Authentic XML Data, 227
Save command, 225
in StyleVision, 225
Saving, 207
HTML to XML conversion, 207
Saving generated files, 30
Saving schema, 187
in HTML to XML conversion, 187
Schema, 173
and Root Template, 173
editing for SPS, 223
editing in mid-session, 223
effect of changing name or location on StyleVision Power Stylesheet, 173
effect of modifying on StyleVision Power Stylesheet, 173
opening to create SPS, 33
saving in HTML to XML conversion, 201
viewing output of HTML to XML conversion, 187
general description, 20
in HTML to XML conversion, 188
Schema Window, 18, 20
description of, 18, 20
symbols and icons, 211
Schemas, 89
importing modules, 89
popular industry standards and SPSs, 4
Schemas and namespaces, 88
Select all, 240
in StyleVision, 240
Selected object in SPS, 301
properties of, 301
Selective processing, 176
of siblings in StyleVision Power Stylesheets, 176
Settings, 294
HTML, 294
Show large markup, 152, 154
Show mixed markup, 152, 154
Show small arkup, 152, 154
Siblings, 176
processing selectively, 176
Spell-checker, 303
in StyleVision, 303
for SPSs, 304
Split cell, 269
horizontally, 269
vertically, 270
SPS, 4
and Authentic View, 4
and document creation process, 4
and Working XML File, 222
auto-adding Date Picker, 284
Block Style in, 21
bookmarks in, 251
closing, 220
conditional templates in, 255
creating, 33
creating from schema, 218
design process, 28
generating XSLT from, 6
hyperlinks in, 252
icons, 211
in document creation process, 3
insert auto-calculation, 253
insert Date Picker, 249
inserting a horizontal line, 246
inserting node contents in, 247
lists in, 245
opening, 219

© 2005 Altova GmbH

SPS, 4
opening schema for, 33
page breaks and numbers, 250
Properties menu for, 295
saving, 33
setting HTML output properties, 296
symbols, 211
table properties in, 272
terminology, 7
Text Style in, 21
uses of, 3
editing dynamic tables, 152
in StyleVision Power Stylesheets, 117
usage of, 158
SPSs, 2
and XSLT, 2
assigning Template XML, 282
closing, 18
for HTML (and PDF in Enterprise), 6
for standard industry schemas, 4
images in, 242
inserting Rest of Contents in, 248
inserting XML tables, 280
paragraph format in, 243
pre-defined formats in, 244
SPSs formatting text and components in, 215
Standard industry schemas, 4
SPS based on, 4
Static (SPS) tables in Authentic View, 158
usage of, 158
Static components/content, 28
Static images, 38
in SPS, 38
in SPSs, 242
Static table, 257
inserting, 257
how to create, 120
how to format, 124
in SPS, 52
in StyleVision Power Stylesheets, 117
joining cells in, 52
modifying properties of ,in SPS, 52
specifying column widths, 52
Static text, 36
in SPS, 36
Styles, 21
in StyleVision, 21
StyleVision, 4
and Authentic View, 4
and document creation process, 4
and tables, 117
Block Style, 21
bookmarks in, 251
closing, 18
closing an SPS, 18
creating lists, 113
data-entry devices, 106
document design, 28
Edit menu, 233
FO compliance levels (Enterprise Edition), 17
formatting in, 215
getting started, 16
How to use..., 86
hyperlinks in, 252
icons, 211
images in, 242
in document creation process, 3
insert auto-calculation, 253
insert Date Picker, 249
Insert menu, 241
inserting a horizontal line, 246
inserting node contents, 247
inserting Rest of Contents, 248
introduction, 1
lists, 245
namespaces in, 88
page breaks and numbers, 250
paragraph format, 243
pre-defined formats in, 244
Redo command, 235
running from the command line, 148
Select all command, 240
setting up, 17
setting up FO processor in Enterprise Edition, 17
symbols, 211
Table menu, 256
terminology, 7
Text Style, 21
Undo command, 234
User Reference, 210
StyleVision features, 8
StyleVision Power Stylesheet, 201
saving in HTML to XML conversion, 201
conditional templates, 133
creating dynamic tables, 121

© 2005 Altova GmbH

StyleVision Power Stylesheet, 201
creating lists, 113
creating static tables, 120
data-entry devices, 106
Date Picker, 136
formatting of numeric fields, 138
input formatting, 138
namespaces in, 88
overview of tables in, 117
table formatting, 124
StyleVision Tutorial, 32
files used in, 32
StyleVisionTutorial, 32
introduction, 32
Support for StyleVision, 319, 320
Symbols, 211
in StyleVision, 211

T

Tab Navigation Buttons, 18
Table, 260
append column to, 260
append row to, 259
delete column from, 264
delete in SPS, 258
delete row from, 263
editing in Authentic View, 285
formatting of ,in SPSs, 272
insert column in, 262
insert row in, 261
inserting a static table, 257
menu in StyleVision, 256
properties, 297
show/hide borders in StyleVision, 271
vertical alignment of cell content, 273
formatting with Block Style Window, 21
Table footers, 73
in SPS, 73
Table properties, 48
cellpadding, 124, 272
in SPS, 52, 59, 73
in StyleVision, 272
Tables, 152
editing dynamic (SPS) tables, 152
formatting in Stylesheet Designer, 211
formatting in StyleVision, 124
headers and footers in PDF, 124
in HTML to XML conversion, 195
inserting in Authentic View, 280
joining cells in, 265, 266, 267, 268
moving in StyleVision Power Stylesheet, 174
overview in StyleVision Power Stylesheets, 117
icons for editing XML tables, 162
usage of, 158
using SPS (static and dynamic) tables, 158
using XML tables, 159
Target namespaces, 89
and <AUTH-STYLE>s, 89
Technical support for StyleVision, 319, 320
Template, 33
saving XSLT files, 33
Template XML file, 7
for SPS, 282
Text, 180
overflow in PDF output, 180
in SPS, 77
in SPSs, 21
in SPS, 44
Text state icons, 278
defining in SPSs, 278
for Authentic View, 77
in SPS, 77
Text Style Window, 18
for formatting in SPS, 44
Text styles, 215
in SPSs, 215
Text View, 32
of Working XML file, 32
Tips, 172
Tool tips, 275
for Authentic View nodes, 275
Toolbar, 26
general description, 26
Tools, 302
in StyleVision, 302
Troubleshooting, 180
Tutorial, 185
HTML to XML conversion, 185
Tutorial StyleVision, 32

© 2005 Altova GmbH

U

Undo, 234
in StyleVision, 234
in HTML to XML conversion, 188, 207
unparsed-entity-uri function, 147
Updating a node, 130
with an Auto-Calculation result, 130
URIs, 147
of unparsed entities, 147
User Info, 275
for Authentic View nodes, 275
in SPS, 81
Using Global Templates, 96

V

Validate XML, 288
in Authentic View, 288
Validation, 275
additional in Authentic View, 275
Vertical alignment of table cell content, 273
in SPSs, 273

W

Working XML file, 7
assigning for SPS, 33
assigning to SPS, 222
print preview, 229
printing, 230
validating in Authentic View, 288

X

XML, 289
convert HTML to, 289
in SPS, 81
XML Document creation process, 3, 4
XML file, 201
saving in HTML to XML conversion, 201
viewing output of HTML to XML conversion, 187
XML Spy web site, 322
XML tables, 126
description of, 126
enabling in StyleVision Power Stylesheets, 126
in StyleVision Power Stylesheets, 117
inserting in SPSs, 280
icons for editing, 162
usage of, 159
XMLSPY, 33
Authentic, 33
XPath, 79
for additional validation, 79
in Auto-Calculation, 59
in conditional templates, 64
version support, 87
for selecting combo box entries, 106
XSL-FO, 17
compliance levels (Enterprise Edition), 17
XSLT, 2
and SPSs, 2
editing XSLT generated from StyleVision, 177
saving in HTML to XML conversion, 201
selecting version, 87
and StyleVision, 2
XSLT stylesheets, 33
from SPS, 33
from SPSs, 6
generating from StyleVision Power Stylesheet, 30
XSLT-for-HTML tab, 24
general description, 24

© 2005 Altova GmbH